CoEdit*
Reference Manual

INTERACTIVE

product family

INTERACTIVE

A Kodak Company

CoEdit*
Reference Manual

This manual provides specific information for using Version 1.00 of CoEdit, an integrated editor.

Contents

Preface: Using This Manual xxi

Chapter 1: Setting Up CoEdit

Overview 1-1
Loading CoEdit 1-1
Command Line Switches 1-2
    Automatic Macro Switch -A 1-2
    Screen Column Switch -C 1-3
    Directory Switch -D 1-3
    Foreign Characters Switch -F 1-3
    ABORT Key Switch -KA 1-4
    Keystroke Typeahead Switch -KK 1-4
    Prefix Key Switch -KP 1-4
    Keyboard Timeout Switch -KT 1-4
    Screen Line Switch -L 1-5
    Other Graphics Switch -O 1-5
Setting Parameters in the Operating System 1-6
Automatic File Saving 1-6

Chapter 2: Defining the CoEdit Keyboard

Key Mapping 2-1
Prefix Key 2-1
Special Keys 2-2

Chapter 2: Defining the CoEdit Keyboard (Cont.)

The ENTER Key 2-2
Overtype Mode 2-2
Insert Mode 2-2
The ABORT Key 2-2
The BACKSPACE Key 2-3
The DELETE Key 2-3
The TAB Key 2-3
The CAPS LOCK Key 2-3
Prompts 2-3
Yes/No Prompts 2-4
Entering Text at Prompts 2-4
Finishing Input at a Prompt 2-4
Help at Prompts 2-4
Special Keys at Prompts 2-5
^A 2-5
^B 2-5
^D 2-5
^F 2-6
^G 2-6
F1 2-6
^L 2-6
^N 2-6
^Q 2-6
^S 2-7
^T 2-7
^U 2-7
^X 2-7
Tab 2-7
Left Arrow 2-7
Right Arrow 2-7
Up Arrow 2-8
Down Arrow 2-8
Ctrl-Left Arrow 2-8
Ctrl-Right Arrow 2-8
Ctrl-End 2-8
PageUp and PageDown 2-8

Chapter 3: The CoEdit Help System

Overview 3-1
Menus 3-1
The On-Line Manual 3-2
Help at Prompts 3-2

Chapter 4: Using the CoEdit Editor

Overview of CoEdit 4-1
CoEdit Command Groups 4-1
    Prompt 4-2
    Message 4-2
    Caution 4-2
    Restriction 4-2
Files 4-3
    File Searching Method 4-3
Windows 4-4
    Types of Windows 4-4
        Subwindows 4-4
    The Window Border 4-5
    The Communication Window 4-6
    Scroll Lock Processing 4-6
    Windows Without Files 4-6
    Commands that Control Files and Windows 4-6
Blocks 4-7
Buffers 4-7
Delimiters 4-8
Undo 4-8
Screen Layout 4-8
    The Status Line 4-9
        File Name 4-9
        Line Number 4-9

Chapter 4: Using the CoEdit Editor (Cont.)

Column Number 4-10
Total Lines 4-10
Text Entry Mode 4-10
Block Type 4-10
Windows Open 4-10
Lock State 4-11
Screen Message Areas 4-11
Entering CoEdit Commands 4-12
Command Mode 4-12
Changing the Binding of Keys to Commands 4-13
Command Keys 4-13
Text Entry 4-13
Overtype and Insert 4-13

Chapter 5: Using the CoEdit Macros

Overview 5-1
CoEdit Macros 5-1
CoEdit Macro Keys 5-1
Macro Files 5-2
Macros in Memory 5-2
Canceling an Executing Macro 5-2
CoEdit Macro Language 5-2
Macro Source File Format 5-3
Function References 5-3
Text and Keystrokes 5-3
Compiling CoEdit Macros 5-3
From Within CoEdit 5-4
From the Operating System Prompt 5-4
Help at coC 5-4
Specifying Other Error Output Files 5-5
Specifying the Output Directory 5-5

Chapter 5: Using the CoEdit Macros (Cont.)

Executing CoEdit Macros 5-5
Debugging CoEdit Macros 5-6
Summary of Macro Debugging Options 5-7
Macro Tokens 5-7
CML Logical Values 5-8

Chapter 6: CML Reference

Overview 6-1
CML #debug 6-2
CML Border_msg 6-3
CML Break 6-4
CML Breakpoint 6-5
CML Cancelmac 6-6
CML Continue 6-7
CML Do 6-8
CML Endmac 6-9
CML Execmac 6-10
CML Foreach_window 6-11
CML GetEnter 6-12
CML GetYN 6-13
CML Goto 6-14
CML If 6-15
CML Label 6-16
CML No_Update 6-17
CML Proc_menu 6-18
CML Prompt 6-20
CML Restart 6-21
CML Update 6-22
CML Varfield 6-23
CML Wait 6-24
CML While 6-25

Chapter 7: Block Commands

Using Block Commands 7-1
Defining the Block 7-1
Inserting Blocks of Text 7-2
    Inserting Horizontally 7-2
        Example 1 7-2
        Example 2 7-3
        Example 3 7-3
    Inserting Vertically 7-4
        Example 7-4
Block All 7-5
Block Begin 7-6
Block Copy 7-7
Block Delete 7-9
Block End 7-11
Block Filter 7-12
Block Lower 7-14
Block Move 7-15
Block Narrow 7-17
Block Program 7-19
Block Quick 7-20
Block Remove 7-22
Block Save 7-23
Block Text Wrap 7-24
Block Upper 7-26
Block Vertical 7-27
Block Widen 7-28
Block Yank 7-29

Chapter 8: Cursor Commands

Using Cursor Commands 8-1
Cursor Movement 8-1
    Movement On the Current Line 8-1
    Movement Between Lines 8-1
    Movement Within the Window 8-1

Chapter 8: Cursor Commands (Cont.)

Cursor Bot Window 8-2
Cursor Down Line 8-3
Cursor End Window 8-4
Cursor Finish Line 8-5
Cursor Home 8-6
Cursor Left Word 8-7
Cursor Mid Window 8-8
Cursor Right Word 8-9
Cursor Start Line 8-10
Cursor Top Window 8-11
Cursor Up Line 8-12
Cursor End Word 8-13
Cursor Left One 8-14
Cursor Right One 8-15

Chapter 9: Delete Commands

Using Delete Commands 9-1
Delete Command Summary 9-1
Delete To Bottom of File 9-3
Delete To Character 9-4
Delete Line 9-5
Delete To End of Screen 9-6
Delete To Finish of Line 9-7
Delete To Home 9-8
Delete Word Left 9-9
Delete One 9-10
Delete Prev 9-12
Delete Word Right 9-13
Delete To Start of Line 9-14
Delete To Top of File 9-15
Delete Undo 9-16
Delete Word 9-18
Delete Yank 9-19

Chapter 10: Environment Commands

Using Environment Commands 10-1
Saving the CoEdit Environment 10-1
co.cfg 10-2
co.nfy 10-2
Menu Definition 10-2
Menu Bindings 10-3
Function Key Bindings 10-3
Environment Attribute 10-5
Environment Configure 10-6
A. Autoindent 10-6
B. Horizontal Scroll 10-6
C. Command Mode 10-7
D. Safety Prompts 10-7
E. Pad ASCII Files 10-7
F. Cursor Top Line 10-8
G. Cursor Bottom Line 10-8
H. Help Menus 10-8
I. Insert Mode 10-9
J. Beep Tone 10-9
K. Duration 10-9
L. Pitch 10-10
M. Move on Yank Line 10-10
N. Find File 10-10
O. Error Directory 10-10
P. Delimiters 10-11
Q. Loc/Rep Ignore Case 10-11
R. Backup Levels 10-11
S. Showmatch 10-12
T. Buffer Characters 10-12
U. Undo Buffers Max 10-12
V. Evaluator Base 10-12
W. New Windows 10-13
X. Expand New Windows 10-13
Y. Yes/No Return 10-13
Z. Clear Screen 10-13
1. Maintain Column Pos 10-14
2. Prototyping 10-14
3. Autosave Mode 10-14
4. Autosave Period 10-14

Chapter 10: Environment Commands (Cont.)

5. Show Side Borders 10-15
6. Show Line Continuation 10-15
Environment Language 10-17
A. Compiler 10-17
B. Error Extension 10-18
C. Command 10-18
Environment Macro 10-20
A. Macro Extension 10-20
B. Macro Library 10-20
C. Automatic Macro 10-20
D. Maximum macro size 10-21
E. Begin Key Scan Code 10-21
F. End Scan Key Scan Code 10-22
G. Default to Library 10-22
Environment Printer 10-24
Environment Read 10-26
Environment Save 10-27
Environment Tab 10-28
Display 10-28
Type 10-28
Width 10-29
Stops 10-29
Justification 10-29
Left Margin 10-30
Right Margin 10-30
Environment Update 10-32

Chapter 11: File Commands

Using File Commands 11-1
File Names 11-1
^F processing at prompts 11-1
Automatic File Finding 11-2
Control-L Processing at File Prompts 11-2
Special Multiple File Processing 11-2

Chapter 11: File Commands (Cont.)

File Another 11-4
File Done 11-5
File Exchange 11-7
File Load 11-9
File Merge 11-11
File Next 11-12
File Open 11-14
File Read 11-15
File Save 11-16
File Update 11-17

Chapter 12: Goto Commands

Using Goto Commands 12-1
Goto Begin Block 12-2
Goto Column 12-3
Goto End Block 12-4
Goto Insert Mode 12-5
Goto Line 12-6
Goto Match 12-7
Goto Overtype 12-8
Goto Previous 12-9
Goto Text Marker 12-10
Goto Window 12-11

Chapter 13: Internal Commands

Using Internal Commands 13-1
Internal Character Value 13-2
Internal Delete Buffers 13-3
Internal Key Code 13-4
Internal Table of Characters 13-5

Chapter 14: Jump Commands

Using Jump Commands 14-1
Jump Back Half Page 14-2
Jump Down Line 14-3
Jump End of File 14-4
Jump Forward Half Page 14-5
Jump Top of Screen 14-6
Jump End of Screen 14-7
Jump Next Page 14-8
Jump Previous Page 14-9
Jump Scroll Lock 14-10
Jump Top of File 14-11
Jump Up Line 14-12

Chapter 15: Key Commands

Using Key Commands 15-1
Key Commands 15-2
Key List 15-3

Chapter 16: Locate Commands

Using Locate Commands 16-1
Entering Locate Options 16-1
    Which 16-2
    Direction 16-2
    Case 16-2
    Lines 16-2
    How 16-3
    The All Option: Find All Mode 16-3
Using Regular Expressions 16-4

Chapter 17: Macro Commands

Using Macro Commands 17-1
Defining a Keystroke Macro 17-1
- Temporary Macros 17-2
- Permanent Macros 17-2
- Ending Keystroke Macro Recording 17-2

CML Macros 17-3
- Defining a CML Macro 17-3
- CML Macro Commands 17-3
- Editing a Macro 17-4
- Executing and Terminating a Macro 17-5
- Searching for Macros on the Disk 17-5
- Changing and Adding Macro Key Bindings 17-5
- Debugging Macros 17-5

Macro Compile 17-6
Macro Define 17-9
Macro End Definition 17-12
Macro Flush 17-13
Macro Go 17-14
Macro Key Name 17-15
Macro List 17-16
Macro Save 17-18
Macro Trace 17-20
Macro Uncompile 17-23

Chapter 18: Print Commands

Using Print Commands 18-1
Print Again 18-2
Print Block 18-4
Print Cancel 18-5
Print Delete 18-6
Print Eject 18-7
Print File 18-8
Print Hold 18-10
Print Pause 18-12
Print Queue 18-14

Chapter 18: Print Commands (Cont.)

Print Release 18-16
Print Terminate 18-17
Print Window 18-18

Chapter 19: Quit Commands

Using Quit Commands 19-1
Quit Save-Exit 19-2
Quit Terminate 19-4
Quit All Windows 19-6

Chapter 20: System Commands

Using System Commands 20-1
System Copy 20-2
System Delete 20-3
System Goto 20-4
System List 20-6
System Rename 20-8
System Type 20-9

Chapter 21: Text Commands

Using Text Commands 21-1
    Description of Tabs in CoEdit 21-1
    Soft Tabs 21-1
        In Overtype Mode 21-2
        In Insert Mode 21-2

Chapter 21: Text Commands (Cont.)

Hard Tabs 21-2
In Overtype Mode 21-3
In Insert Mode 21-3
Fixed Tabs 21-3
In Overtype Mode 21-3
In Insert Mode 21-3
Text Tab Guide 21-4
Text Center Line 21-5
Text Date 21-6
Text Dup Line 21-7
Text Dup One Character 21-8
Text Quote 21-9
Text Remove Marker 21-10
Text Set Marker 21-11

Chapter 22: Verify Commands

Using Verify Commands 22-1
Verify Compile 22-3
Verify File Reset 22-5
Verify Line Reset 22-6
Verify Next Error 22-7
Verify Previous Error 22-8
Verify Read Error 22-9
Verify Syntax 22-11
Verify Tag 22-13

Chapter 23: Window Commands

Using Window Commands 23-1
Windows on the Screen 23-1

Chapter 23: Window Commands (Cont.)

Opening and Closing Windows 23-2
Moving Between Windows 23-2
Window Attributes 23-2
The Window Commands 23-3
Window Assign 23-5
Window Bury 23-6
Window Compare 23-7
Window Copy 23-9
Window Expand 23-11
Window Hide 23-12
Window List 23-13
Window Move 23-14
Window Next 23-16
Window Open 23-17
Window Previous 23-18
Window Quit 23-19
Window Read-Only 23-20
Window Size 23-21
Window Tab 23-22
Window Undo 23-23
Window Visible 23-25
Window Slide 23-26

Chapter 24: Xternal Commands

Using Xternal Commands 24-1
Xternal Chmod 24-2
Xternal Execute 24-3
Xternal Make Directory 24-4
Xternal Remove Directory 24-5
Xternal Shell 24-6
Xternal Touch 24-7

Chapter 25: Yank Commands

Using Yank Commands 25-1
Summary of Yank Commands 25-2
Yank To Bottom of File 25-3
Yank To a Character 25-4
Yank Line 25-5
Yank To End of Screen 25-6
Yank To Finish of Line 25-7
Yank To Home 25-8
Yank Word Left 25-9
Yank Word Right 25-10
Yank To Start of Line 25-11
Yank To Top of File 25-12
Yank Word 25-13

Chapter 26: Miscellaneous Commands

Using Miscellaneous Commands 26-1
Misc Again 26-2
Misc Back Tab 26-3
Misc Changed 26-4
Misc Date 26-5
Misc Evaluate 26-6
Misc Open Above 26-8
Misc Ins/Ovr 26-9
Move Right 26-10
Misc New Line 26-11
Misc Open Below 26-12
Misc Paint Screen 26-13
Misc Repeat 26-14
Misc Sound 26-15
Misc Tab 26-16
Misc Undo Delete 26-17
Misc Execute 26-18
Misc Enter-Auto 26-19
Misc Enter 26-20

Appendices

Appendix A: CoEdit Commands and Macros A-1
Appendix B: CoEdit Messages B-1
Appendix C: Filters C-1
Appendix D: Stream Editing D-1

Index

Preface: Using This Manual

The *CoEdit Reference Manual* describes the CoEdit editor. It is recommended that you read through this manual before using this product.

This manual provides information describing the use of CoEdit in both the UNIX* and XENIX* operating system environments. Within this guide, the name UNIX or the words "operating system" are used when referring to either UNIX or XENIX commands.

Intended Audience

This manual is written for programmers who may already be familiar with a text editor, such as vi or emacs.

A working knowledge of the UNIX operating system is also recommended. For additional information on the UNIX operating system, refer to *The UNIX Programming Environment* by Brian W. Kernighan and Rob Pike, Prentice Hall Inc.

Organization of Information

This manual consists of 26 chapters, four appendices, and an index.

Chapter 1 describes loading CoEdit and command line switches.

Chapter 2 describes the CoEdit keyboard.

Chapter 3 describes the CoEdit on-line help system.

Chapter 4 describes how to use CoEdit command groups, screen layout, and text entry.

Chapter 5 describes the use of CoEdit macros and the CoEdit Macro Language (CML).

Chapter 6 provides a reference to the CoEdit Macro Language.

Chapters 7 through 26 provide information about CoEdit commands. Each chapter presents a command group (for example, Block, Delete, and Cursor commands). The chapters are presented in alphabetical order according to the name of the command group.

Within each chapter, the individual commands that are part of the command group are described. (Individual commands are presented in alphabetical order according to keystrokes.)

Appendix A describes CoEdit commands and macros.

Appendix B describes filters that can be written to work with CoEdit.

Appendix C lists error messages provided by CoEdit.

Appendix D describes stream editing with CoEdit.

The index provides a quick reference for finding important CoEdit terms.

Syntax Conventions

The following syntax conventions are used in this manual.

1. Typewriter font is used for examples, such as the following:

   If the macro you wish to execute is called "STARTUP" then entering the following will cause that macro to execute immediately.

   ```
   co -A:STARTUP
   ```

2. Variable information is printed in italics, for example,

   The Directory switch format is -D=*path*, where *path* is the name of the directory.

3. When the control key is pressed at the same time as another key, then the control key is represented by a caret (^). For example, ^B

indicates that the control key is pressed at the same time as the letter B. This may also be written as Ctrl-B.

4. In general, this manual cannot describe CoEdit for all terminals. Whenever you see a mention of specific keys, be aware that they may not be the same on your terminal. Refer to your terminal user's guide or terminfo guide for more information. (See Chapter 2 for a discussion of the CoEdit keyboard.)

Chapter 1: Setting Up CoEdit

Overview 1-1
Loading CoEdit 1-1
Command Line Switches 1-2
- Automatic Macro Switch -A 1-2
- Screen Column Switch -C 1-3
- Directory Switch -D 1-3
- Foreign Characters Switch -F 1-3
- ABORT Key Switch -KA 1-4
- Keystroke Typeahead Switch -KK 1-4
- Prefix Key Switch -KP 1-4
- Keyboard Timeout Switch -KT 1-4
- Screen Line Switch -L 1-5
- Other Graphics Switch -O 1-5

Setting Parameters in the Operating System 1-6
Automatic File Saving 1-6

Chapter 1: Setting Up CoEdit

Overview

This chapter describes how to load and format CoEdit. (See your release notes for information on installing CoEdit.)

Loading CoEdit

CoEdit can be loaded from the operating system prompt by typing co and then pressing ENTER.

A sign-on screen will be displayed during program initialization, and the main CoEdit screen will appear. The cursor will be placed in an empty window, ready for you to create a new file or load one from the disk.

It is possible to load one or more files automatically from the command line by typing the file name(s) to load after co. For example:

```
co main.c ovl1.c ovl2.c ENTER
```

Filenames that use wildcard characters (* and ?) are loaded as if they had been entered at a File Read command; as many of the files matching the pattern are loaded into windows as can fit in available memory (see Chapter 11 concerning File Commands).

```
co *.c /usr/src/asm/*.asm ENTER
```

A directory name may be specified on the command line. CoEdit will read in all the files from that directory.

```
co /asm ENTER
```

There is no limit to the number of filename parameters that may be specified on the command line, except that which is imposed by UNIX, which is dependent on your system configuration.

Command Line Switches

CoEdit's command line switches are used to pass information to CoEdit as it loads. They are preceded by a switch character indicating to CoEdit that the parameter is a switch. For UNIX, the switch character is a dash (-). The switch character is followed by a letter, in upper- or lower-case. Generally speaking, they take the following form

```
-XY
```

where X is the switch itself, and Y is any switch-dependent information that may be required. If the switch supplies information to CoEdit, the information must immediately follow the letter or be separated by a colon or equal sign (with no intervening spaces). For example, the following are all valid:

```
-E25      -e:25      -E=25
```

Invalid and ill-formed switches are ignored with no message. In the event that a switch is specified more than once, only the last occurrence of the switch will be considered.

Automatic Macro Switch -A

The "A" or Automatic macro switch allows you to specify a macro that is to be executed immediately upon program execution. This switch has two main applications.

First, the specified macro can be used to execute some specific commands every time you begin editing. The second application is to provide a means for stream editing. (See Appendix D for a discussion of stream editing.)

All command line switches and parameters are processed as usual before execution of the automatic macro. The macro to be executed is named immediately after the -A switch.

For example, if the macro you wish to execute is called "STARTUP", then entering

```
co -A:STARTUP
```

will cause that macro to execute immediately.

If the specified macro does not exist, CoEdit will load normally, as though the -A switch were not actually on the command line.

Screen Column Switch -C

The -C or screen Column switch tells CoEdit how many columns in width your monitor is. For example,

```
co -C132
```

would tell CoEdit that you have 132 columns on your screen. (VGA cards can support this.)

Note that this switch does not reset or reprogram your video adapter; it merely tells CoEdit to assume that width. The minimum value is 80; there is no maximum. The default is 80.

Directory Switch -D

The D or Directory switch changes CoEdit's default master directory. The format of the Directory switch is -D=*path*, where *path* is the name of the directory. CoEdit uses this directory to locate the help files and as a temporary work directory. If no directory is specified, the default master directory is /usr/bin/LPI. If this directory does not exist when CoEdit is started, then CoEdit will create it. (If CoEdit needs to create the directory, then it will delete that directory upon program termination.)

Foreign Characters Switch -F

The -F or Foreign Characters switch allows the use of foreign characters in CoEdit. CoEdit uses some characters in upper range ASCII for special functions. By default, they are in the range of 128 through 141, where the foreign characters are. By indicating -F or -F1, they will be moved to 224 through 237. -F2 will move them to 240 through 253. You may also move them to another location (above 127) by indicating -F*nnn* where *nnn* is a number from 128 through 242.

ABORT Key Switch -KA

The -KA switch defines the ABORT key for the current session of CoEdit. By default, the ABORT key is the Esc key. (See Chapter 2 for a discussion of the keyboard.) To use the switch, you indicate the ASCII value of the key you want to be the ABORT key. For example, to make the ABORT key Ctrl-D, you would use co -KA4. To make it #, you would enter co -KA35. (For an explanation as to why you may need to change this value, see the -KT switch section later in this chapter.)

Keystroke Typeahead Switch -KK

CoEdit normally senses keyboard typeahead (characters entered in advance of CoEdit's ability to process them). This allows CoEdit to optimize screen output. However, this can occasionally cause CoEdit to mis-draw the screen. This switch, -KK, tells CoEdit not to look for typeahead, with the effect that screen updating might be less efficient, but it will always be correct.

Prefix Key Switch -KP

The -KP switch defines the prefix key for this session of CoEdit. (See Chapter 2 for a full description of the prefix key.) With this option, you may set the prefix key or disable it. For example,

```
co -KP
```

enables the prefix key and sets it to its default value, which is ^R (that is, Ctrl-R).

-K. disables the prefix key altogether. -K*x* enables the prefix key and sets its value to Ctrl-*x*, where *x* is a letter from A to Z.

Keyboard Timeout Switch -KT

The -KT switch tells CoEdit how long to wait before aborting an attempt to identify a keyboard escape sequence. Normally this value need not be changed from its default setting of 1 second.

Occasionally during heavy system activity, especially if you are operating a terminal over a network, an escape sequence sent by a keystroke on

your terminal might be sent in two or more sections to CoEdit. If the amount of time between these sections is more than the timeout period, CoEdit will fail to recognize the escape sequence, with the result that characters will be sent to CoEdit in ways that you had not intended (for example, into the middle of your text).

You can use the -KT switch in conjunction with the -KA switch to alter the ABORT key for CoEdit. If the ABORT key is Esc (default), you will notice a one-second delay (or whatever timeout period you have set) before CoEdit reacts to it, because CoEdit has to make sure that Esc is not the beginning of a terminal escape sequence. If, however, you make the ABORT key something with which no escape sequences from your terminal begin, you will experience no delay in the processing of the ABORT key.

To change this value, you specify to CoEdit a number representing 1/10th second increments. The default value is 10 (that is, 1 second). The maximum value is 100 (that is, 10 seconds), though a value of 0 will cause CoEdit to wait indefinitely. For example, to change the timeout to 2 seconds, use -KT20.

Screen Line Switch -L

The -L is the screen Line switch that tells CoEdit how many lines you have on your monitor. For example,

```
co -L60
```

would tell CoEdit that you have 60 lines on your screen. (VGA cards can support this.) Note that this switch does not reset or reprogram your video adapter; it merely tells CoEdit to assume that length. The minimum value is 25; there is no maximum. The default is 25.

Other Graphics Switch -O

This switch tells CoEdit that it can use upper-range ASCII graphic characters on the screen. By default, CoEdit cannot assume that graphic characters are compatible with your terminal.

Setting Parameters in the Operating System

You may set CoEdit parameters in the operating system environment. CoEdit will then examine the environment upon loading so that you do not have to specify the switches each time the program is loaded. You can put the appropriate statement into your startup configuration file (for example, .profile or .cshrc) or enter it at the operating system prompt.

For example,

```
setenv co -Astartmac -D=/lpi/CoEdit   (csh)
co -Astartmac -D=/lpi/CoEdit; export co (sh)
```

Any combination of the switches may be specified in any order. Switch flags on the command line take precedence over their equivalents in the environment.

Automatic File Saving

The automatic file saving feature in CoEdit allows you to work continuously without worrying about losing data because of power failures or system crashes.

To use automatic file saving, do the following:

First, you must tell CoEdit that file saving should be triggered either by elapsed time or by number of keystrokes.

Second, CoEdit needs to know the amount of elapsed time or the number of keystrokes between file saves. These are options 3 and 4 in the Environment Configure command. (Refer to Chapter 10 for more information on this command.)

CoEdit then saves your work in a subdirectory called co. This directory is created in your home directory when the first automatic file save is performed. All subsequent automatically saved files are placed in that directory. When your CoEdit program terminates, those files will be deleted and the directory will be removed.

Automatically saved files do not have the same names as the actual files, so that there are no conflicts and no windows are UNNAMED. (See Chapter 23 for a discussion of the Windows commands.) Therefore,

names assigned to those automatically saved files consist of the file's CoEdit window number (four digits) followed by the extension .csf for CoEdit Save File. The first line of the file will contain the real path and file name. The rest of the file is an exact duplicate of the actual file, reflecting your work as of the last automatic file saving.

If you start CoEdit when there are autosave files present, CoEdit will ask you if you want to restore them, ignore (that is, delete) them, or terminate CoEdit without touching them.

If you choose to restore them, the files will be read and available for editing. The first line of the restored file is the full pathname of the original file. Simply deleting this line will restore the file to its last edited state. Any file names specified on the command line where CoEdit was started are ignored, but command line switches are interpreted.

If you choose to ignore the autosave files (by entering No at the restore prompt) the automatic save files will be deleted.

If you terminate CoEdit (by entering Terminate at the restore prompt), the program will terminate immediately and the automatic save files will be untouched.

Chapter 2: Defining the CoEdit Keyboard

Key Mapping 2-1
Prefix Key 2-1
Special Keys 2-2
The ENTER Key 2-2
Overtype Mode 2-2
Insert Mode 2-2
The ABORT Key 2-2
The BACKSPACE Key 2-3
The DELETE Key 2-3
The TAB Key 2-3
The CAPS LOCK Key 2-3
Prompts 2-3
Yes/No Prompts 2-4
Entering Text at Prompts 2-4
Finishing Input at a Prompt 2-4
Help at Prompts 2-4
Special Keys at Prompts 2-5
^A 2-5
^B 2-5
^D 2-5
^F 2-6
^G 2-6
F1 2-6
^L 2-6
^N 2-6
^Q 2-6
^S 2-7
^T 2-7
^U 2-7
^X 2-7
Tab 2-7
Left Arrow 2-7
Right Arrow 2-7
Up Arrow 2-8
Down Arrow 2-8
Ctrl-Left Arrow 2-8
Ctrl-Right Arrow 2-8
Ctrl-End 2-8
PageUp and PageDown 2-8

Chapter 2: Defining the CoEdit Keyboard

Key Mapping

The keys F1, PageUp and PageDown appear frequently within this manual, as do the Macro Begin and End Definition keys. These keys have been mapped to F1, F2, F3, F9, and F10, respectively, for use under UNIX on a VT100* terminal.

Note that these keys are machine-specific and may not apply to your machine or keyboard. If you are using a terminal that does not have F1, F2, F3, F9, or F10, or your keyboard does not map the VT100 keys directly, then you can remap these key bindings by using the environment key file and by using the Environment Macro command (see Chapter 5).

In general, we cannot describe CoEdit for all terminals. Whenever you see a mention of specific keys, be aware that they may not be the same on your terminal. Refer to your terminal user's guide or terminfo guide for more information.

Prefix Key

Some keyboards may not always support all keystrokes. For example, the ^I key is indistinguishable from the TAB key, ^S and ^Q may toggle output to the screen, and ^M is no different than RETURN.

To get around this problem, CoEdit uses a prefix key. This prefix key, defined by default to be ^R (that is, Ctrl-R), tells CoEdit that the next keystroke should be treated as a control character. Therefore, if you wanted to execute the command Macro Compile (normally ^MC), you would enter the three keystrokes ^RMC (Ctrl-R followed by the letter M, followed by the letter C).

This prefix key is only active when CoEdit is waiting for command input, and not at prompts. The prefix key may be disabled or changed using the provided command line option (see Chapter 1). If it is disabled, or it is not the default ^R, then the default key binding for ^R (if any) will be active.

Special Keys

Some keys on the keyboard have special meaning during text entry. Some of them have two meanings, depending upon the current text entry mode, or may perform differently depending on where the cursor is in the current line. With a little experimentation you will soon be familiar with the way these keys work.

The ENTER Key

The ENTER key, also known as the RETURN key, is generally used to place an ENTER character in the current line. The ENTER character signals the end of a line.

Overtype Mode

The ENTER key will move the cursor to the beginning of the next line in the window. If the cursor is on the last line in the window, it will create a new line and move the cursor to it.

Insert Mode

The ENTER key will create a new line after the current line and move all characters, if any, that were at or after the cursor onto this new line. This will effectively split the old line into two new ones at the old cursor position.

The ABORT Key

The ABORT key is the key that is used to clear input, go back to the previous prompt (if any), cancel a command, or let CoEdit know that you are done viewing information, all depending upon the situation.

The BACKSPACE Key

The BACKSPACE key deletes the character to the left of the cursor and shifts all characters from the current cursor position to the end of the current line one character left. If the cursor is at the beginning of a line, the BACKSPACE key will delete the ENTER character at the end of the previous line and join the two lines together.

The DELETE Key

This key deletes the character under the cursor and shifts all characters to the right of the cursor left by one. If the cursor is at the end of the current line, DELETE deletes the ENTER character (which ends every line, except the last one) and the next line in the file is joined to the current one. If the cursor is on a TAB character, the text to the right of the cursor may shift more than one position.

The TAB Key

The TAB key moves the cursor to the next tab stop. The way in which the TAB key works depends on the text entry mode. (See Chapter 21 for a complete discussion of tabs.)

The CAPS LOCK Key

The CAPS LOCK key controls whether or not the letters A-Z are automatically put into uppercase as they are typed. It works very much like a typewriter. When CAPS LOCK is on, the keys a-z generate uppercase letters. To enter a lowercase letter, press Shift and the letter. When CAPS LOCK is off, the keys A-Z generate lowercase letters; to enter an uppercase letter, press Shift and the letter.

Prompts

When you execute a command that requires input, CoEdit opens the communication window at the top of the screen and displays one or more prompts. (A prompt is a short message describing the type of input requested.) The cursor will be placed in the communication window and CoEdit will wait for your input. This section describes different prompts and how you can respond to them.

Yes/No Prompts

Many commands display prompts that require a Y or N response. Some of these will display a default response. Pressing ENTER accepts the default response. You may only press Y or N at these prompts; all other characters are ignored (if environment option BEEP is on, CoEdit will beep). If environment option Yes/No Return is set to On, CoEdit will wait for you to press ENTER after selecting a yes or no response. If Yes/No Return is set to Off, CoEdit will accept your response immediately without requiring an ENTER.

Entering Text at Prompts

Text entry at prompts is very similar to text entry during editing. If CoEdit is in insert mode, characters will be inserted, and if CoEdit is in overtype mode, characters will be overwritten. The BACKSPACE and DELETE keys work just as they do during editing.

If a default response is presented on the screen, you may accept it simply by pressing ENTER. Or, if you begin typing right away, the default response will be deleted and replaced by your input. The default response may be retrieved by typing ^D (see the section "Special Keys at Prompts" later in this chapter). Note that each prompt has an input field of a certain length; you may not be able to insert characters into the field if it is already full.

Finishing Input at a Prompt

There are two ways to leave a prompt. The first is to enter and edit the data requested and press ENTER, thereby signalling completed input.

The second is to press ABORT when the input field is blank. This tells CoEdit to return to the previous prompt and allow re-entry of the data there. If ABORT is pressed when the first prompt's input field is blank, the command is canceled.

Help at Prompts

Help is provided at prompts in two ways. First, as a time-saving feature, CoEdit will provide default strings at many prompts. For many commands, the default string is the string last entered at the prompt.

See the section "Special Keys at Prompts" later in this chapter for information on using the default string.

On-line help is also provided at prompts through the use of messages on the border. If you press F1, CoEdit will display a series of messages offering suggestions on how to input the requested information. These will appear on the border of the communication window. When no more border messages are available, CoEdit may offer more extensive help by opening a window below the communication window and displaying one or more windows of help. Use the PageUp and PageDown keys to scroll through this additional help, and press ABORT when you are ready to resume input. (See Chapter 3 for additional information about the help system.)

Special Keys at Prompts

In addition to the normal editing keys (BACKSPACE, DELETE) and the keys mentioned previously (ENTER, ABORT, ^F1), the following keys perform special functions when the cursor is at a prompt:

^A

Appends the default string onto the input field and moves the cursor to the end of the field.

^B

Blanks the input field. (Erases input.)

^D

Replaces the current input with the default string and moves the cursor to the start of the field.

^F

On some file prompts, ^F displays a list of files in one or more directories. Files shown in protected mode (low-lighted) are read-only. Files shown with an asterisk after their names are hidden. (See Chapter 11 for further information concerning File Commands.)

^G

Grabs the current word (as defined by the CoEdit delimiters) and puts it in the response. This is useful for eliminating typing responses that appear on the screen (search texts, etc.).

F1

Displays help for the current command. Help will be displayed first on the border and subsequently in the entire screen.

^L

Searches for the named file on your computer. See Chapter 10 for information concerning the environment option FINDFILE.)

^N

Loads the input area with the name of the file in the current window. If the current window has no name, CoEdit will beep.

^Q

On some print prompts, displays the the print queue. (See Chapter 18 for further information concerning the Print Commands.)

^S

On prompts requiring a filename, ^S selects the file that is currently highlighted by the ^F key. For more information on the File commands, see Chapter 11.

^T

Allows you to enter control characters into the input area. For example, to enter ^S, type ^T^S.

^U

Will undo the first line of the first delete buffer into the input area. This works exactly the same as the Miscellaneous Undo command. For more information, see Chapter 26.

^X

Transposes the two characters before the cursor.

Tab

Inserts the special tab character (~) as used by the Locate commands into the text.

Left Arrow

Moves the cursor one character left.

Right Arrow

Moves the cursor one character right.

Up Arrow

Moves the cursor up one line.

Down Arrow

Moves the cursor down one line.

Ctrl-Left Arrow

Moves the cursor to start of field.

Ctrl-Right Arrow

Moves the cursor to end of field.

Ctrl-End

Clears the input area from the cursor rightward.

PageUp and PageDown

Scrolls file, key, queue, window or help display if one of these is active. (These keys may be keyboard-specific. Check your terminal manual or terminfo guide to learn which keys on your terminal correspond to PageUp and PageDown.)

Chapter 3: The CoEdit Help System

Overview 3-1
Menus 3-1
The On-Line Manual 3-2
Help at Prompts 3-2

Chapter 3: The CoEdit Help System

Overview

CoEdit has an on-line help system to give you instant help while you are in an editing session. This chapter describes the help system and how to use it. There are three basic entry points into the system:

- the help menus
- the on-line manual
- help at prompts

Menus

CoEdit will display a menu for a specific command group if you pause more than a few seconds after pressing the key that identifies the group (for example, ^K for Key Commands). The menu is temporarily opened at the top of the screen; the window will be restored after a menu selection has been made or command entry has been canceled.

Menus for command groups (with the exception of the F1 help menu) will not be displayed if the Environment Configure option Help has been turned Off. However, if you press a question mark (?) after the command's group letter, the menu will be displayed.

Menus consist of one or more command names displayed with the letters that invoke them. When a menu is displayed, the first item in the menu is highlighted. A one-line description of the item appears below the menu. You can use the arrow keys or the spacebar to move the highlighted area from one item to the next.

There are several ways to leave a menu. You may press the letter identifying the command you wish to select, or you may press ENTER to select the item currently highlighted. Either method will select the command. If you press ABORT, you will cancel the command selection completely.

If you have redefined the keys on the keyboard, it may be possible that a command group will have no commands in it. In that case, CoEdit will display the message "Menu is empty" on the border and emit a beep.

The On-Line Manual

Information presented in the on-line documentation that comes with CoEdit duplicates the information in the CoEdit manual that describes CoEdit commands and command groups. The on-line documentation is accessed through the Help command, executed by pressing F1. (Because CoEdit users have different terminals, F1 is the key on your keyboard that maps to the CoEdit-defined F1. See Chapter 2 for further information on CoEdit keys.)

When you press F1, a menu is displayed immediately. This menu lists all of the command groups, each identified by the first letter of its name. Below the menu, a window is opened with the first page of General Help. This window will display one or more pages of help for each command group and each command in CoEdit.

To see the help for a command group, press the first letter of its name. The menu for that group is displayed at the top of the screen, and the general help for the group will appear in the window below. To see the help for a specific command in a command group, press the first letter of that individual command's name and view its help in the window. To execute the command whose help is displayed, press the F1 key. If you press ABORT, you will return to the General Help menu.

To scroll the contents of the help window, use the keys on your keyboard that map to the CoEdit-defined PageUp and PageDown keys (see Chapter 2 for further information on CoEdit keys).

To return to editing, press ABORT until the screen is restored and the cursor reappears.

Help at Prompts

In addition to command menus and the on-line manual, CoEdit offers help at many of the prompts in the program. If you are at a prompt and you need assistance, press F1. If help is available at that particular prompt, it will appear on the border of the communication window. Keep pressing F1 to see more help.

Some prompts have more extensive help, in which case a window will open below the communication window and help will be displayed. The PageUp and PageDown keys will scroll the contents of the help window. To continue with the command, press ABORT to return to the command's prompt.

Chapter 4: Using the CoEdit Editor

Overview of CoEdit 4-1
CoEdit Command Groups 4-1
- Prompt 4-2
- Message 4-2
- Caution 4-2
- Restriction 4-2

Files 4-3
- File Searching Method 4-3

Windows 4-4
- Types of Windows 4-4
  - Subwindows 4-4
- The Window Border 4-5
- The Communication Window 4-6
- Scroll Lock Processing 4-6
- Windows Without Files 4-6
- Commands that Control Files and Windows 4-6

Blocks 4-7
Buffers 4-7
Delimiters 4-8
Undo 4-8
Screen Layout 4-8
- The Status Line 4-9
  - File Name 4-9
  - Line Number 4-9
  - Column Number 4-10
  - Total Lines 4-10
  - Text Entry Mode 4-10
  - Block Type 4-10
  - Windows Open 4-10
  - Lock State 4-11
- Screen Message Areas 4-11

Entering CoEdit Commands 4-12
- Command Mode 4-12
- Changing the Binding of Keys to Commands 4-13
- Command Keys 4-13

Text Entry 4-13
- Overtype and Insert 4-13

Chapter 4: Using the CoEdit Editor

Overview of CoEdit

CoEdit was designed to be fast, to provide the opportunity for multiple file/window editing, and to provide a complete development environment that is linked to assemblers and compilers, highlighting lines and displaying error messages. CoEdit was also designed to provide basic system commands, as well as the ability to move among directories and projects without leaving the program.

To make it easier for the user to remember how to use CoEdit commands, command names correspond to the keys that access the command. For example, to print a file, the command is ^PF, where the caret symbol (^) represents the control key. CoEdit also allows you to redefine the keyboard. (See Chapter 2 for further information concerning CoEdit keys.)

All the commands are divided into command groups. For example, there are commands to move the cursor, quit the program, print a file, or locate a string. The command groups into which these commands fall are Cursor, Quit, Print, and Locate, respectively. There are twenty-one command groups and over 200 commands in all.

CoEdit Command Groups

The command group chapters in this manual (Chapters 7 through 26) are presented alphabetically by the first letter of the command group name. Within each chapter, individual commands are described, each starting on a new page. (Individual commands are presented alphabetically according to the keystrokes entered for the command.) The key sequence and command name are shown in the banner at the top of the page and a description of the command is presented, followed by one or more of the following sections:

Prompt

Lists the prompt(s) which the command may display to request input from you. Each prompt listed will have a corresponding sentence or two describing in detail what is required.

Message

Lists the message(s) that the command may display to inform you of error or other conditions. Each message listed has a description of how to correct the error or explains in more detail what the message means.

Caution

Lists any cautions that the command may display. Cautions include questions such as "Window has been modified. OK to quit?" They are usually warning of some potential data loss that would result from abandoning an edited window or overwriting a file on the disk.

Restriction

Lists the situations under which this command is restricted from use. Some commands cannot be used when the current window is read-only (such as deletes, window copies or moves). Many commands cannot be used if the current window is in a "find all" level (see Chapter 16 concerning Locate Commands). The situations listed are those under which the command is restricted (that is, may not be used).

For example, if the restriction is "Read Only", that means you may not use that command when you are in a Read-Only window. The restrictions are described in the following table:

RESTRICTION	MEANING
Find All	In Find All mode
Not	Not in Find All mode
Read Only	Current window is Read Only
Defining	When recording a macro
Block	Current window is "Narrow"
Not	Current window is not "Narrow"

Files

A file is a set of related data stored as one unit on a mass storage device. CoEdit's purpose is to create new files or modify the contents of existing ones. This version of CoEdit (the UNIX version) uses the operating system's virtual memory, and is therefore limited only by your system's swap space.

File Searching Method

When CoEdit requests a file name from you, you may have CoEdit look for that file in your disk directories. This is useful if you cannot remember where a file is located, or if you simply do not care to type the directory name yourself. CoEdit will take the name you have entered into the input area and look for it. Of course, the file name must not contain the wildcard characters [,], *, or ?.

You may have CoEdit look for files in one of two ways. CoEdit will automatically look for the file if you set the environment variable Find File simply by pressing the ENTER key after the file name, as you normally would. You may also press ^L (that is, Control-L, with the L representing Locate) before you press ENTER. This will make CoEdit go out and look for the file on the disk. CoEdit will then display the file name with the directory it found, or with no directory if the file was not found.

CoEdit will search for the file in the following manner:

- First, it will try to find the file as you entered it.

  If it does not exist, and the file name you entered begins with a backslash (\), then the search ends and the file was not found.

- If the file name you entered has an extension, CoEdit will look in the operating system environment for a variable name that matches the extension. If it exists, it should contain one or more directory names separated by colons (exactly as the operating system PATH variable does). For example, if you enter main.c and in the operating system environment there is the following environment variable

  c=/usr/work/c:/usr/work/lib/c

  then CoEdit will look first in

/usr/work/c and then in /usr/work/lib/c.

- If CoEdit cannot find the file in any of those directories, then it will use the operating system PATH environment variable to search for the file. If CoEdit still cannot find the file, the search ends and the file is not found.

Windows

A window is a specific section of the screen through which its contents, usually a file, are viewed. However, a window does not have to be associated with a file. For example, if you simply open a window (using Window Open - see Chapter 23 concerning Window commands) and start typing, that window is not yet associated with a disk file. Only when the contents of the window are saved to a disk, or a disk file is read into a window, is the window then associated with a file. You can make a window any size to control how much of the contents of the window you can see at one time.

Many windows can be displayed simultaneously on a CoEdit screen, each surrounded by a border. The border of the current window is highlighted; the cursor always rests in the current window, and there is always a current window open. (See the section "Window Border" later in this chapter for further information.)

Types of Windows

CoEdit has two types of windows: normal windows and subwindows. Normal windows are those windows that contain the complete contents of their associated disk file, or everything you have created prior to saving to disk. Subwindows are those windows that contain only a subset of the complete text in a window.

Subwindows

There are two types of subwindows in CoEdit:

- The Find All window is created by the Locate String "all" and "not all" options, which gather lines that match (or lines that do not match) into a subwindow.

- The Block Narrow window is created by the Block Narrow command, which makes a subwindow out of the text in the current block.

Certain commands are restricted from use in these subwindows, but otherwise their text can be edited as in a normal window.

The Window Border

Each window border contains information in the upper left corner that provides the following:

- the window number, which can be used to identify the window at prompts
- the short name of a window, if defined
- the file name associated with the window
- the data format of the window (see Chapter 23 concerning the Window Format command), which is left blank for ASCII files
- the Find All level displayed in square brackets ([]) (see Chapter 16) and the Block Narrow level displayed in braces ({}) (see Chapter 7)
- an asterisk immediately after the window number, if the window has been changed

For example, the following window border:

```
=0013*SAVE FOO.C(R)(E)[03]{02}====
```

tells us that the name of the file in the window is FOO.C. The file has been assigned an internal CoEdit name of SAVE. It is read-only, and it is EBCDIC. It is in Block Narrow level three, and Find All level two. In addition, the window contents have been modified (before it was made read-only).

The following is a typical window border:

```
=0001 LISP.PAS====================
```

This window border indicates that this is the first window and it contains the unchanged file LISP.PAS.

The Communication Window

When you execute a command that requires input, CoEdit opens a special window at the top of the screen called the communication window. One or more prompts is displayed and the cursor moves into this window. When input is complete, the communication window is removed and the screen is restored to its previous status. (See the section "Entering Text at Prompts" in Chapter 2 for information on entering data at prompts.)

Scroll Lock Processing

Usually, commands only affect the current window. It is possible, however, to make the Jump commands affect all windows currently on the screen that are not obscured, by pressing the key associated with the Scroll Lock, or by turning Scroll Lock On by way of the Jump Scroll command (see Chapter 14). When Scroll Lock is On, the word Lock will appear on the status line. In this way you can move through several files simultaneously.

Windows Without Files

It is possible to open a window without specifying the name of the file you wish to edit. In this kind of window the name on the border will be UNNAMED. This is the name of the window that appears when CoEdit is loaded without any file names on the command line. Unless you supply a filename, you cannot save the contents of this window onto the disk, as you will be prompted for a filename when you ask to save the file.

Commands that Control Files and Windows

The File and Window command groups contain the commands that read and save files into and out of windows, move the cursor from one window to another, and move text between windows. (See Chapter 11 concerning File Commands, and Chapter 23 concerning Window Commands for more information on these commands.)

Blocks

During text entry, you can delete, move, or insert blocks of text. CoEdit has columnar (that is, vertical) blocks as well as horizontal blocks. A horizontal block is defined as all characters (including ENTER and TAB characters) located between the block begin marker and the block end marker.

A columnar block is one that contains the characters between the begin and end markers, but only within the limits of the columns they are in. For example, if the begin marker is in column 1 on line 12 and the end marker is in column 9 of line 20, then the block will include all characters between columns 1 and 8 (inclusive) on lines 12 through 20 (inclusive).

These special markers are displayed on the screen using the greater-than (>) and the less-than symbols (<). They are inserted into your text with the Block Begin and Block End commands.

Once a block is defined, you can perform several commands to manipulate it. A block can be deleted, yanked, moved, and copied; it can be uppercased and lowercased; and it can be printed and saved to the disk. A block can also be passed as input to an external program and replaced with the program's output. The commands that perform these functions are located in the Block command group (see Chapter 7).

Buffers

When a block or any other unit of text is deleted or yanked, it is not lost but is rather saved in a buffer file.

CoEdit saves a certain number of delete buffers containing deleted text in case you want to "undo" a deletion (see Misc Undo in Chapter 26, Delete Undo in Chapter 9, and Window Undo in Chapter 23).

The number of deletions saved can be set in the environment (see Chapter 10 concerning Environment Commands). You can also set whether or not the single characters deleted by DELETE and BACKSPACE should be buffered.

The delete buffers can themselves be removed with the Internal Delete Buffer command. Removing one or all buffers will free swap space for editing.

Delimiters

Delimiters in CoEdit are those characters that separate words in the file. You may set the characters that CoEdit will consider to be delimiters in the environment.

For example, Cursor Right Word and Cursor Left Word, and Delete Word are just a few of the commands that use the set of delimiters. The most obvious delimiter is a space. We have chosen what we feel to be a good set; however, you should feel free to add or delete delimiters according not only to your editing style but also to what you edit. By using CoEdit, you can best determine what delimiters are right for you and your environment.

Undo

The word undo in CoEdit has two different meanings:

1. The most frequent and important meaning of undo relates to deleted text. When you undo deleted text in CoEdit, you recover it from CoEdit's memory of deletions, and put it into the current window at the current cursor position. The commands to do this are Miscellaneous Undo, Delete Undo, and Window Undo. (See Misc Undo in Chapter 26, Delete Undo in Chapter 9, and Window Undo in Chapter 23).
2. The second meaning of undo refers to the "Find All" windows described in Chapter 16. To undo a Find All window is to replace all of the lines that were found and so to make the Find All window into the parent window. In this way, the Find All operation is undone.

Screen Layout

During normal editing, the CoEdit screen has two major areas: the status line and the editing area. The status line, which is the line at the top of the screen, displays important information about the current state of the editor and some of the keyboard keys. The editing area below the status line can contain one or more windows. Each window in the editing area is surrounded by a border separating it from the other windows. Messages and other information will often appear on the current window's border.

The current window, the one in which the cursor resides and which most commands affect, is displayed with its border highlighted.

The Status Line

The single line at the top of the screen displays important information about the current window and the state of the editor in general. The status line information refers only to the current window, except as noted below. Here is a sample status line (displayed on two separate lines in order to fit onto this printed page):

```
File: VIDEO.C          L: 193  C: 25
Lines: 294    Ins  H    0: 04    Lock
```

From left to right, the items on the status line are the following:

File Name

The file name of the current window is the first item in the status line. Only the name and extension of the file are displayed; to see the full path name, use the Window List command. If the current window has no name, it will be called UNNAMED. In the preceding example, the file name is VIDEO.C.

This item will change as any of the following occurs:

- you move from one window to the next
- the current window is given a new file name (that is, after a disk save or read command)
- a new window is opened

Line Number

The current line number appears after the symbol L:. This is the line number on which the cursor sits. Lines in a window are numbered sequentially from 1 to the total number of lines in the window. This item will change as you move the cursor from line to line.

Column Number

The current column number appears after the symbol C:. This is the column number on which the cursor sits. It is the current position on the current line, where column 1 is the first character in the line. This number will be updated constantly as the cursor moves.

Total Lines

The total number of lines appears after L: in the current window.

Text Entry Mode

The current text entry mode is displayed between the total lines item and the word Open as follows:

Ins	if the editor is in insert mode
Ovr	if the editor is in overtype mode
Cmd	if the editor is in command mode

(See the section "Text Entry" later in this chapter for further discussion of insert and overtype modes.)

Block Type

The current block type is displayed just before the O: and shows an H if blocks are currently treated as horizontal, or a V if they are currently vertical (columnar).

Windows Open

The number of windows currently open appears after the O:, including hidden windows. Use the Window List command to see a full list of open windows.

Lock State

If Scroll Lock is On, the word Lock appears after the shift state indicator. When Lock is displayed, the Jump commands are simultaneously executed in all non-obscured windows on the screen. This allows you to page or scroll easily through separate files to visually check their similarity.

Screen Message Areas

CoEdit has an extensive set of messages, which include prompts, warnings, informational and error messages, and helpful hints. There are only a few locations on the screen where messages appear.

Most messages appear on the border of the current window, at the top right corner. Many commands will display error messages here. For example, a Locate command might display "Text not found" on the current window border if it could not find the text you were looking for. The Locate Replace command might display "15 changes made" to inform you of its successful completion. During input in the communication window, input error messages will also appear on the border at the top right corner.

During normal editing, messages displayed on the border will be removed after a few keystrokes. It is possible to clear this message area during editing by pressing the ABORT key.

A second message area exists on the status line at the very left of the screen. This area is most often used to display the message "Wait" during operations that may require more than a few seconds to complete. It is also used to display the command group name when you press a control key. For example, if you press ^B (that is, Ctrl-B), the word Block will appear in this area. This is to remind you that a second letter is required to identify the specific command.

In addition, this area is used to indicate that CoEdit is recording a keystroke macro. If the word MACRO flashes there, all of your keystrokes are being captured for later replay. (See Chapters 5 and 6 for more information on macros.)

Another message area is used in connection with the Verify commands and with macro debugging. It is possible that messages will appear on the bottom border of the current window. (See Chapter 22 concerning

Verify Commands, and Chapters 5 and 6 concerning macros, for further information on the commands that use this area.)

Entering CoEdit Commands

Commands are instructions to CoEdit, telling it what to do with the text you have entered.

For example, if you wish to save a few lines of text on your disk or diskette, you need to enter a command instructing CoEdit to perform this operation. There are several commands that will save text in CoEdit. For example, you can issue a File Save. To enter this command, you would hold down the Control key (marked Ctrl on the left of the keyboard) and press F at the same time.

Pressing a Control key such as Control-F (also written as Ctrl-F or ^F) identifies the command group in which your selected command is located. (Note that it is possible to change the key that identifies a command group to an Alt-key such as Alt-F if your keyboard has an Alt key. See Chapter 10 concerning Environment Commands.)

To execute the actual command, you would then press the first letter of its name. Thus ^F followed by S will execute the File Save command. Many commands, including File Save, will require input from you. In this case, CoEdit needs to know the name of the file in which your text should be stored. See the section "Special Keys at Prompts" in Chapter 2 for information on making responses when CoEdit requests input.

Command Mode

It is possible to put CoEdit into command mode, a special mode that turns normal text entry Off. The Environment Configure command (see Chapter 10 concerning Environment commands) will allow you to toggle command mode On or Off. When command mode is On, the word Cmd appears where the text entry mode usually appears on the status line and commands no longer have to be entered with Ctrl or Alt held down. In command mode, pressing the letter f will access the file commands and not insert (or overtype) the letter f. Use the Environment Configure (see Chapter 10 concerning environment commands) command to return to normal text entry. (Note that it is impossible to enter the letters of the alphabet when you are in command mode.)

Changing the Binding of Keys to Commands

CoEdit has over 200 commands, which have been carefully named and assigned to groups in a logical fashion. It is possible, however, to alter the way the commands are organized. See Chapter 10 concerning Environment Commands, for information on restructuring the command groups to suit your needs.

Command Keys

It is possible to execute a command without typing two letters (as in ^FS = File Save). Almost all of the keys on your keyboard can be assigned to a command such that pressing one of these keys executes the command in one step. These keys can be given "meanings" (see Chapters 5, 6, and 17 for more information concerning macros, as well as Chapter 15 concerning Key Commands). They can all be defined during an editing session (see Chapter 10 concerning Environment Commands).

If you wished to execute a series of one or more commands with one keystroke, you would create a macro. For an introduction to macros, see Chapters 5 and 6.

Text Entry

As soon as you load the program, a cursor appears on the screen indicating where the first character you type will be placed. You can then press alphanumeric and symbol keys on the keyboard and they will appear on the screen. (Note that some keys on the keyboard have special meanings. See the section on "Special Keys" in Chapter 2.)

Overtype and Insert

While you are entering text with CoEdit, there are two text entry modes. The first is overtype. In this mode, CoEdit acts very much in the same way as a typewriter. As you press each character, it types over the character that was at the cursor, thus replacing it. The cursor then moves one character to the right.

The second mode is insert. In this mode, CoEdit does not replace the character at the cursor with the one you typed. Instead, it moves all of the characters from the cursor to the end of the line over, and then

inserts the character you pressed. After the character is inserted, the cursor moves one character to the right. In this way you can insert multiple characters into the middle of a line without manually making room for them.

The current text entry mode is displayed on the status line at the top of the screen. Ins means CoEdit is in insert mode; Ovr means CoEdit is in overtype mode.

Chapter 5: Using the CoEdit Macros

Overview 5-1
CoEdit Macros 5-1
    CoEdit Macro Keys 5-1
    Macro Files 5-2
    Macros in Memory 5-2
    Canceling an Executing Macro 5-2
CoEdit Macro Language 5-2
    Macro Source File Format 5-3
    Function References 5-3
    Text and Keystrokes 5-3
    Compiling CoEdit Macros 5-3
        From Within CoEdit 5-4
        From the Operating System Prompt 5-4
        Help at coC 5-4
    Specifying Other Error Output Files 5-5
    Specifying the Output Directory 5-5
Executing CoEdit Macros 5-5
    Debugging CoEdit Macros 5-6
    Summary of Macro Debugging Options 5-7
    Macro Tokens 5-7
CML Logical Values 5-8

Chapter 5: Using the CoEdit Macros

Overview

This chapter describes CoEdit macros and the CoEdit Macro Language (CML). (For more information on CML macros, see Chapters 4 and 17.)

CoEdit Macros

A macro is a disk file that contains a series of commands and keystrokes. Macros are used to store sets of commands that are used frequently or are very long and repetitious. When a macro is executed, these commands and keystrokes are replayed as if you were typing them in yourself. After CoEdit reaches the end of the macro, you are then free to enter keystrokes as usual.

Because macro files can invoke other macro files, or even invoke themselves, macros can loop and repeat a set of keystrokes hundreds or thousands of times automatically.

While a macro is executing, keys pressed during macro execution will be stored until after the macro is done. When the macro has finished executing, CoEdit will then process those keys. If the macro executes another program, the program will be able to read those keystrokes.

CoEdit Macro Keys

Macros in CoEdit may be bound to almost any key sequence. This means that the keystroke represents the macro. For example, if the macro FINDALL is bound to Ctrl-Q, then pressing Ctrl-Q will execute the macro FINDALL. Generally speaking, any keystroke (aside from alphanumeric keys such as A, and +) listed in the section "Macro Tokens" later in this chapter may have a macro bound to it.

Macro Files

Macros are stored on your disk when you define them and are read from the disk when they are executed. Macro file names consist of the macro name and the extension .cmf. The name of a macro is specified when it is defined or compiled. The macro file extension is configurable. (See Environment Configure in Chapter 10 for more information.)

When you press the key to which that macro is bound, CoEdit searches the current working directory for the macro file. If if cannot be found, CoEdit searches the current macro library directory. This library directory can be set in the environment (see Chapter 10). If the macro file cannot be found in either directory, the message "Macro file missing for this key" is displayed.

Macros in Memory

CoEdit will store macro files in memory after their first execution to speed up subsequent executions. However, not all macros will be stored. The macros to be stored in memory must be small enough to warrant having them in memory. (If they are large, they will take longer to execute, and having them in memory will not enhance their execution speed anyway.) The size of the macros that will be stored in memory can be set in the environment (see Chapter 10).

Canceling an Executing Macro

To cancel a macro that is executing, press Ctrl-Break or press the Alt and Ctrl keys simultaneously until the macro terminates. The message "Macro execution canceled" will be displayed and CoEdit will begin accepting keystrokes from the keyboard again.

CoEdit Macro Language

This section describes the CoEdit Macro Language (CML). CML allows you to create more sophisticated macros using Do, For, While, and If programming strutures to test conditions and create program loops.

There are also CML keywords and control statements that are used in compiling and executing the macros. (See Appendix A.)

Macro Source File Format

Each of the first two lines of every CML source file must start with an asterisk (*), as in the following example. The first line must consist of the name of the macro. The macro name can be up to eight characters, all of which must be valid operating system file name characters. The second line must consist of a description of the macro. For example,

```
*INSTAB
*Insert a tab and move down one line
```

Function References

In CML you may access CoEdit editing functions directly by name. For example,

```
Go_down_line
```

is the name of the function that moves the cursor down one line. Macro function names are listed in Appendix A.

Text and Keystrokes

Normal text and keystrokes can be entered in CML source as well as CoEdit commands. Text should be enclosed in quotation marks. When the macro executes, CoEdit interprets the quoted text as if it had been typed at the keyboard. Single keys can be entered in the source file using names listed in Appendix A. For example, to code the keypad home key, the macro source would contain the word Home.

Compiling CoEdit Macros

CoEdit macros can be compiled either from within CoEdit or from the OS prompt. Macro files have the extension .cms for source files and .cmf for object files.

From Within CoEdit

CoEdit has an external compiler for its macro language and understands the CML compiler in the same way that it understands other language compilers. This means that you can compile right from within CoEdit and have the compiler errors highlighted on the screen. To use this function from within CoEdit, just use the Macro Compile or the Verify Compile commands.

From the Operating System Prompt

You can also compile CML macros from the operating system command line.

The compiler is named coC for CoEdit Macro Compiler. If you have a source macro file named doit in the current directory, simply typing

```
coC doit
```

will compile the file doit.cms. coC will automatically put any compiler errors or warnings generated during the compilation into the file doit.err. coC will, however, look in the CoEdit configuration file CoEdit.cfg for the error directory and the default error file extension.

These two pieces of information (the error directory and the default error file extension) are configured with the Environment Configure command, Option O, and the Environment Language command, language M, option B, respectively. If they are supplied, then coC will put the files in the appropriate directories. If they are not present, or if the CoEdit configuration files cannot be located, coC will place the resulting files into the current directory.

Help at coC

To get help from coC regarding command line flags, just type

```
coC -?
```

This will cause coC to display a screen of help. coC can also uncompile your macros; that is, it can translate the macro executable object code back into compilable source. This can be done using the -u command line flag, as in the following example:

```
coC -u doit
```

In this situation, coC will read the doit.cmf file and create a new doit.cms file. The new doit.cms file may then be edited and recompiled.

This feature of coC allows the user to delete macro source files and yet still be able to retrieve them.

Specifying Other Error Output Files

The -ef flag may be used to indicate that the error and warning information should be written to a different file. Any disk file name may be specified, or either of the following standard devices:

- stdout
- stderr

For example:

```
coC -efstderr doit
```

Specifying the Output Directory

The -ed flag may be used to specify a different directory for error files; this overrides the error directory found in the configuration file. The following example places the error files for doit in the /usr/joe directory.

```
coC -ed/usr/joe doit
```

Executing CoEdit Macros

Macros may be executed in any of three ways. First, you may bind macros to keys by simply editing the CoEdit.key file. (See Chapters 10 and 17 for further information.)

Second, macros may be executed by using the Miscellaneous Execute command (^ZX), which will prompt you for the name of a macro to execute and then execute that macro. (See Chapter 26 concerning Miscellaneous Commands.)

Finally, macros may be executed from other macros by either generating the key sequence to which the macro is bound, or by invoking the macro by name with the Execmac macro function. (See Chapter 6 for further information.)

Debugging CoEdit Macros

CML has a source level debugger built into CoEdit, with which you can trace the execution of a macro, delete commands, insert keystrokes, and trap breakpoints.

In order to debug a macro, the macro must have been compiled using the #debug coC compiler directive or by specifying -d on the coC command line. This causes debugging information to be included in the object macro file. Then you must issue the Macro Trace command to enable macro debugging in CoEdit. This way you can debug only the macros you want, while you let the other macros run freely.

When you debug a macro, a debugging window appears on the screen. The source to the macro appears on the bottom border of the window. For example, the following line might appear:

```
0003:If Validblock - F
```

This means that you are viewing source from line three of the macro source file; the source for that line is If Validblock; and that the value of the condition (Validblock) is False (that is, there is not a valid block defined).

In addition to this information, a pop-up debugging menu appears on the screen. It presents you with the debugging options, and you need to choose one of these options when the debugging window appears. The next section summarizes the macro debugging options.

Summary of Macro Debugging Options

OPTION	DESCRIPTION
G	Go. This means that the macro may continue untraced. Control of CoEdit will return to the keyboard when macro execution terminates, or when the macro encounters a breakpoint.
Space	Trace single instruction. This lets the current instruction execute.
ABORT	Terminate. This terminates macro execution immediately.
Ins	Insert keystroke. By hitting the Insert key, you are able to insert keystrokes into the macro (but only for this execution).
Del	Delete instruction. Hitting the Delete key causes CoEdit to skip the current instruction and pass on to the next.
Up	View previous instruction. By hitting the Up arrow on the keyboard, you can view the previous instruction.
Dn	View next instruction. By hitting the Down arrow on the keyboard, you can view the next instruction.
I	(Same as Ins.)
D	(Same as Del.)
-	(Same as Up.)
+	(Same as Dn.)

Macro Tokens

These are the symbols or tokens representing the keys as used by the Macro Compile and the Macro Uncompile commands. When compiling macros, these symbols may be in uppercase or lowercase.

All individual characters (ASCII 0 through hex FF) are entered as the characters themselves. The only exceptions are the double quote (") and the backslash (\), which are preceded by a backslash.

Alt-	Alt0	Alt1	Alt2
Alt3	Alt4	Alt5	Alt6
Alt7	Alt8	Alt9	Alt=
AltA	AltB	AltC	AltD
AltE	AltF	AltF1	AltF2
AltF3	AltF4	AltF5	AltF6
AltF7	AltF8	AltF9	AltF10
AltG	AltH	AltI	AltJ
AltK	AltKey	AltL	AltM
AltN	AltO	AltP	AltQ
AltR	AltS	AltT	AltU
AltV	AltW	AltX	AltY
AltZ	Backspace	Back-Tab	CtrlA
CtrlB	CtrlC	CtrlD	CtrlE
CtrlEnd	CtrlF	CtrlF1	CtrlF2
CtrlF3	CtrlF4	CtrlF5	CtrlF6
CtrlF7	CtrlF8	CtrlF9	CtrlF10
CtrlG	CtrlH	CtrlHome	CtrlI
CtrlJ	CtrlK	CtrlL	CtrlLeft
CtrlM	CtrlN	CtrlO	CtrlP
CtrlPgDn	CtrlPgUp	CtrlPrtScn	CtrlQ
CtrlR	CtrlRight	CtrlS	CtrlT
CtrlU	CtrlV	CtrlW	CtrlX
CtrlY	CtrlZ	Del	Dn
End	Enter	Esc	F1
F10	F2	F3	F4
F5	F6	F7	F8
F9	Home	Ins	Keypad5
Left	PgDn	PgUp	PrntScrn
Right	Shift0	Shift1	Shift2
Shift3	Shift4	Shift5	Shift6
Shift7	Shift8	Shift9	ShiftF1
ShiftF2	ShiftF3	ShiftF4	ShiftF5
ShiftF6	ShiftF7	ShiftF8	ShiftF9
ShiftF10	Tab	Up	

CML Logical Values

CML logical values can be used in If and While statements. They may be preceded by the Not logical negation. For example, if the expression Not After_right is True, then the cursor is not right of the right margin.

LOGICAL VALUE	TRUTH CRITERION
After_right	The cursor is right of the right margin.
Ascii	The current window is ASCII.
Autoindent	Autoindent is On in the environment.
Autosave	Automatic file saving is On in the environment.
Beep	Beep Tone is On in the environment.
Before_left	The cursor is left of the left margin.
Block_begin_line	The cursor is on the line containing the Block Begin marker.
Block_end_line	The cursor is on the line containing the Block End marker.
Bof	The cursor is at the beginning-of-file (i.e., at the beginning of the first line).
Bol	The cursor is at the beginning of the current line.
Bos	The cursor is at the beginning of the screen (i.e., at the home position).
Cancelled	The last executed command was cancelled by the user. This can only be used immediately following execution of the command and before the execution of the next command.
Changed	The contents of the current window have been changed.
Clearscreen	Clear Screen is On in the environment.
Command	CoEdit is in Command Mode.
Compileok	The last program compiled via CoEdit was compiled successfully.
Confirmed	The user entered "Y" at the last Prompt command, and False if the user entered "N". If no Prompt command has been executed, Confirmed is False.
Deflib	Default to Library is On in the environment (Environment Macro command, Option G).
Delchars	True is Buffer Characters is On in the environment.
Eof	The cursor is at the end-of-file.
Eol	The cursor is at the end of the current line.
Eos	The cursor is at the end of the screen.
Expanded	Expand New Windows is On in the environment.
Findall	CoEdit is in Find All mode.
Findfile	Find File is On in the environment.

LOGICAL VALUE	TRUTH CRITERION
Firstline	The cursor in on the first line of the file.
Found	The last attempted Locate String, Next, Again, or Replace command found the source pattern.
Inblock	The cursor is within a defined block.
Insert	CoEdit is in insert mode.
In_margins	The cursor in within the bounds of the margins.
In_scroll_lock	CoEdit is in forced Scroll Lock mode.
Insert	CoEdit is in insert mode.
Isdelim	The current character is a delimiter.
Isdigit	The current character is a digit.
Isexpanded	The current window is expanded.
Isletter	The current character is a letter.
Islower	The current character is lowercase.
Isspace	The current character is white space.
Istab	The current character is a tab.
Isupper	The current character is uppercase.
Justication	Justification is On in the current window.
Lastline	The cursor is on the last line of the file.
Lrignorecase	Ignore Case is on in the environment.
L_Ada	The current file is Ada.
L_Assembly	The current file is Assembly.
L_Basic	The current file is BASIC.
L_C	The current file is C.
L_Cobol	The current file is COBOL.
L_Cxx	The current file is C++.
L_Dbase	The current file is dBase.
L_Eql	The current file is EQL.
L_Fortran	The current file is FORTRAN.
L_Modula2	The current file is Modula-2.
L_Undefined	The current language is undefined.
L_Pascal	The current file is Pascal.
L_PMS	The current file is PMS.
L_Sql	The current file is SQL.
Maintpos	Maintain Column Position is On in the environment.
Morefiles	There are more files available matching the last File Open or File Read command file pattern.
Moveyank	Move on Yank Line is On in the environment.
Narrow	The current window is narrow (see the Block Narrow command in Chapter 7).
Nomem	CoEdit is out of memory.

LOGICAL VALUE	TRUTH CRITERION
Padfile	Pad ASCII Files is On in the environment.
Prototyping	Prototyping is On.
Safety	Safety prompts are On.
Showmatch	Showmatch is On in the environment.
Unix_regexp	CoEdit is using UNIX-style regular expressions.
Unnamed	The current window contains no named file.
Updating	Screen updating during macro execution is On.
Validblock	There is a valid block defined in the current window.
Ynenter	Yes/No Return is On in the environment.

Chapter 6: CML Reference

Overview 6-1
CML #debug 6-2
CML Border_msg 6-3
CML Break 6-4
CML Breakpoint 6-5
CML Cancelmac 6-6
CML Continue 6-7
CML Do 6-8
CML Endmac 6-9
CML Execmac 6-10
CML Foreach_window 6-11
CML GetEnter 6-12
CML GetYN 6-13
CML Goto 6-14
CML If 6-15
CML Label 6-16
CML No_Update 6-17
CML Proc_menu 6-18
CML Prompt 6-20
CML Restart 6-21
CML Update 6-22
CML Varfield 6-23
CML Wait 6-24
CML While 6-25

Chapter 6: CML Reference

Overview

This chapter lists and describes macros found in the CoEdit Macro Language (CML).

CML macros are presented in this chapter in alphabetical order. Each macro explanation starts on a new page and includes a synopsis and description, as well as examples and references to other sections where applicable.

For further information concerning use of CoEdit macros, see Chapter 4.

CML

#debug

Synopsis

```
#debug
```

Description

This compiler directive tells the CML compiler to include macro debugging information in the object macro file. This will allow source level debugging of the macro. (See Chapter 4 for more information on debugging macros.) This directive must be placed immediately after the first two statements, which define the macro name and the macro description.

Example

```
*STARTUP
*Start up macro
#debug
```

Synopsis

```
Border_msg "message"
```

Description

Displays a message on the border of the current window. The message must be enclosed within double quotes and contain zero or more characters.

Example

```
if not found
     Border_msg "That was not found"
endif
```

This example would display a border message if the boolean Found was false.

CML

Break

Synopsis

```
Break
```

Description

Transfers control of execution to the statement beyond the current loop. In other words, it terminates the current program loop.

Example

```
While not eol
     While isspace
          Go_right
          If istab
             Break
          Endif
       Endwhile
       ; The Break transfers control here
Endwhile
```

This Break statement in this macro would transfer control to the comment.

See Also

CML Continue and Chapter 4.

Synopsis

```
Breakpoint
```

Description

Set a source level macro breakpoint. When control is transferred to a Breakpoint statement, the macro debugger will be activated.

See Also

Chapter 4.

CML

Cancelmac

Synopsis

```
Cancelmac
```

Description

Cancels execution of all macros and returns control to the keyboard.

See Also

CML Endmac and Chapter 4.

Synopsis

```
Continue
```

Description

Transfers control of execution to the terminating statement in the current loop (the Loop statement), thereby skipping all intervening statements.

Example

```
Do
     if Block_begin_line
          Continue
     endif
     Del_word
Loop While Not Eof   ; Continue comes here
```

In this example, the Continue statement transfers control to the Enddo statement. Unlike the Break statement, Continue will allow the loop to continue execution if the condition of the loop is met.

See Also

Chapter 4.

CML

Do

Synopsis

```
Do . . . Loop While condition
```

Description

Begins a Do/Loop While loop. *condition* is a CML logical value described in Chapter 5.

Example

```
File_open "*.c" Enter
do
     execmac replace
loop while Morefiles
execmac replace
```

This macro code would execute the macro REPLACE for all .C files in the current directory.

See Also

Chapter 4.

Synopsis

Endmac

Description

Terminates the current macro and transfers control to either the calling macro or the keyboard, as appropriate.

See Also

CML Cancelmac and Chapter 4.

CML

Execmac

Synopsis

Execmac *macro-name*

Description

Executes another macro as subroutine.

Example

```
Execmac TEMP
```

This would execute the macro TEMP. When TEMP was through executing, control would return to the current macro. Note that the macro name is case-sensitive.

See Also

Chapter 4.

Synopsis

```
Foreach_window . . . Endfor
```

Description

Begins a loop whose body will be executed for each open window that is not hidden; therefore it is possible to exclude windows from the limits of the loop.

Example

```
Foreach_window
     Goto_line "450" Enter
     Go_begin_line
Endfor
```

This macro would move the cursor to the beginning of line 450 in each open window.

See Also

Chapter 4.

CML

GetEnter

Synopsis

```
Get_Enter
```

Description

Reads a single ENTER key from the keyboard. All other keys are ignored. CoEdit will ignore this keystroke (that is, it will not be inserted into the text).

Example

```
"Press Enter to continue: "
Get_Enter
```

CoEdit will display the prompt and wait for an ENTER key to be pressed once.

See Also

CML Prompt, CML GetYN, CML Border_msg and Chapter 4.

Synopsis

```
Get_Enter
```

Description

Reads either a Y or an N (uppercase or lowercase) from the keyboard. All other keys are ignored. CoEdit will ignore this keystroke (that is, it will not be inserted into the text).

Example

```
"Do you wish to continue: "
GetYN
if not Confirmed
     Cancelmac
endif
```

CoEdit will display the prompt and wait for a Y or N response. If the user enters N, all macros will be terminated.

See Also

CML Prompt, CML Get_Enter, CML Border_msg, CML Confirmed and Chapter 4.

CML

Goto

Synopsis

```
Goto label-name
```

Description

Transfers control of execution to a CML label statement. *label-name* is the name of a label defined with a Label statement.

Example

```
if not found
     Goto skip
endif
Label skip
```

This macro will transfer control to the label statement. A Goto may reference labels before or after the Goto statement.

See Also

CML Label and Chapter 4.

Synopsis

If *condition* **. . . Else . . . Endif**

Description

Begins an If/Else/Endif statement. *condition* is a CML logical value described in Chapter 4.

Example

```
If Findall
     do
          Locate_undo
     while Findall
else
     border_msg "You're not in Find All mode"
Endif
```

This macro will undo all Find Alls if CoEdit is in find all mode; otherwise it will inform the user that they were not in find all mode.

See Also

Chapter 4.

CML

Label

Synopsis

```
Label label-name
```

Description

Defines a statement label.

Example

```
Label NextOne
```

This statement defines the label "NextOne". Note that labels are case-sensitive; for example, the label "NextOne" is not the same label as "nextone".

See Also

Chapter 4.

Synopsis

```
No_Update
```

Description

Causes CoEdit not to update the screen status line during macro execution. This is useful when using a slow-speed terminal.

See Also

CML Update and Chapter 4.

CML

Proc_menu

Synopsis

```
Proc_Menu
```

Description

Causes CoEdit to accept only the Up arrow, Down arrow, and Enter keys. This allows you to create menus on the screen and have the user move the cursor around to select an option.

Example

```
    Window_open
Label Menu
      "A.* Open a file" Enter
      "B.* Copy a block" Enter
      "C.* Quit CoEdit" Enter
      Go_up_line Go_up_line Go_up_line
      Proc_Menu
      Go_up_line
      If Firstline File_open Goto Cont Endif
      Go_up_line
      If Firstline Block_copy Goto Cont Endif
      Go_up_line
      If Firstline Quit_terminate Goto Cont Endif
      "Invalid Selection" Goto Menu
Label Cont
```

This sample code creates a new window and a small menu in that window. It then calls Proc_Menu to allow the user to make a selection. By using the "Go_up_line" command with the Firstline conditional, the user can determine where the cursor was when they hit Enter.

See Also

Chapter 4.

CML

Prompt

Synopsis

Prompt *prompt-message*

Description

Pauses the macro and gets a response from the user. This function displays a prompt on the border of the current window and gets a Y or N response. The response can be queried with the Confirmed conditional. It will be True if the user entered Y, and false if they entered N. The value of Confirmed will be maintained until the next Prompt statement is executed. The user may terminate the macro at the Prompt statement by entering ^C.

Example

This example will delete the current line if the user enters Y.

```
Prompt "Delete this line?"
if Confirmed
     Del_line
endif
```

See Also

CML Confirmed and Chapter 4.

Synopsis

```
Restart
```

Description

Restarts the current macro by transferring control to the first statement in the macro. This is equivalent to defining a label at the beginning of the macro and later coding a Goto statement referring to that label.

See Also

Chapter 4.

CML

Update

Synopsis

```
Update
```

Description

Causes CoEdit to update the screen status line during macro execution. This reverses the effect of No_Update.

See Also

CML No_Update and Chapter 4.

Synopsis

```
Varfield
```

Description

Pauses macro execution until the Enter key is struck. This function can also be encoded as Alt= "V".

Example

```
File_open
Varfield
Window_assign "WORK" Enter
```

This code will invoke the function File Open, pause until the operator enters a file pattern, open the specified file, and name it WORK.

See Also

Chapter 4.

CML

Wait

Synopsis

```
Wait
```

Description

Pauses macro execution for about five seconds. This may be used to preserve the screen image for viewing before resuming executing. This function can also be encoded as Alt= "W".

See Also

Chapter 4.

Synopsis

```
While condition ... Endwhile
```

Description

Begins a While/Endwhile loop. *condition* is one of the CML logical values described in Chapter 4.

Example

```
While not istab
     if eol
         break
     endif
     " "
     Go_left
Endwhile
```

The code in this example will insert a space between every character on a line until it hits a tab character or the end of the line.

See Also

Chapter 4.

Chapter 7: Block Commands

Using Block Commands 7-1
Defining the Block 7-1
Inserting Blocks of Text 7-2
    Inserting Horizontally 7-2
        Example 1 7-2
        Example 2 7-3
        Example 3 7-3
    Inserting Vertically 7-4
        Example 7-4
Block All 7-5
Block Begin 7-6
Block Copy 7-7
Block Delete 7-9
Block End 7-11
Block Filter 7-12
Block Lower 7-14
Block Move 7-15
Block Narrow 7-17
Block Program 7-19
Block Quick 7-20
Block Remove 7-22
Block Save 7-23
Block Text Wrap 7-24
Block Upper 7-26
Block Vertical 7-27
Block Widen 7-28
Block Yank 7-29

Chapter 7: Block Commands

Using Block Commands

In CoEdit, a block is defined as one or more characters (including carriage-return characters) between the block begin marker and the block end marker.

Blocks in CoEdit may be either horizontal or vertical (see Chapter 4 for a complete description of blocks). Blocks can be moved to the current cursor location from another part of the file, copied to the current cursor location, deleted from the file, or copied to a stack of buffers (see Chapter 9 for a description of Delete commands). Blocks can also be written to a disk file, printed, converted to uppercase or lowercase, and narrowed.

The Block commands only affect the current window and each window has its own block markers. Blocks can be moved or copied between windows with the Window Move, Window Copy, and Window Undo commands. (See Chapter 22 for a description of Window commands.)

Defining the Block

To define the current block, set the block markers with the Block Begin (^BB) and Block End (^BE) or the Block All (^BA) commands. These commands will insert special graphics characters into your text (greater-than and less-than symbols) to mark the begin and end locations. The block markers will never overtype what is under the cursor; they are always inserted. These markers do not remain at a fixed position, and will move when a new character is inserted.

Note that the begin marker must be located before the end marker in the file. You can, however, set the end block marker before you set the begin block marker.

You can remove the markers by using either the Block Remove (^BR) command or the Delete commands. (See Chapter 9 for a description of Delete commands.) You cannot, however, overtype the markers in text

entry mode. If you want to jump to a block marker already set in the current window, you can use the Goto Beg Blk and the Goto End Blk commands (see Chapter 12).

To block off a complete line (including the carriage return at its end), place the begin marker before the first character on the line and the end marker before the first character on the next line. If the end marker is placed at the end of a line, the ENTER character on that line is not part of the block.

Inserting Blocks of Text

You can insert blocks of text in either of the two following ways:

1. horizontally; that is, you can move text to the right of the current cursor location
2. vertically; that is, you can move columns of text from one position to another

Inserting Horizontally

When a block of text is inserted into a file, it will always be placed at the current cursor location. All text to the right of the cursor (the tail) on the current line is temporarily saved and the first line of the block is added at the cursor.

If there is more than one line in the block, the remaining lines are inserted. The tail of the original line is then added back to the file. If the last line in the block included an ENTER character, the tail is added as a new line. If the last line in the block did not include an ENTER, the tail is appended to that line. Three examples of block inserts follow.

Example 1

Insert:

```
end of block line 1 ENTER
line 2
```

Into:

```
Sample line (cursor at 'l' in line)
```

Result:

```
Sample end of block line 1
line 2line (cursor at 'l' in line)
```

The cursor will remain where it was, now at the 'e' of end.

Example 2

Insert:

```
block line 1 ENTER
line 2 ENTER
```

Into:

```
Sample line (cursor at 'l' in line)
```

Result:

```
Sample block line 1
line 2
line (cursor at 'l' in line)
```

Example 3

Insert:

```
**
```

Into:

```
Sample line (cursor at 'l')
```

Result:

```
Sample ** line (cursor at 'l')
```

Inserting Vertically

You can also insert blocks of text vertically into your file; that is, you can move columns of text from one position to another by inserting new text before them, as in the following example:

Example

Insert:

```
12345
67890
abcde
fghij
```

Into:

```
ABCDE          (with cursor at the A)
FGHIJ
KLMNO
PQRST
```

Result:

```
12345ABCDE     (with cursor at the A)
67890FGHIJ
abcdeKLMNO
fghijPQRST
```

The cursor will remain where it was, now at the '1'.

Note that most Block commands require memory allocation calls. See your operating system manual for information on system memory management.

The remainder of this chapter describes individual Block commands.

Description

Sets block markers at the beginning and end of the file in the current window. (Any beginning and ending markers that had already been set will now be removed from their positions.)

Prompt

None.

Message

```
Line would be too long
```

CoEdit has a maximum line length of 254 characters. If the current line is 254 characters long, no markers can be added to it.

Block ^BB

Begin

Description

Sets the block begin marker at the current cursor location. If a begin marker had already been set, it will now be moved from its previous location.

Prompt

None.

Message

```
Line would be too long
```

CoEdit has a maximum line length of 254 characters. If the current line is 254 characters long, no markers can be added to it.

Description

Copies the text in the currently defined block in the current window to the cursor location.

Prompt

None.

Message

```
Block undefined or out of order
```

Begin and end markers must be set in the current window, begin before end, with the end marker to the right of the begin marker for vertical blocks.

```
Line would be too long
```

CoEdit has a maximum line length of 254 characters. If the block was copied to the current cursor location, a line longer than 254 characters would result.

See Also

Window Copy (Chapter 22) for information on copying text blocks between windows.

Restriction

Find All, Read Only

Description

Deletes the text in the current block and adds it to a stack of deleted buffers.

Prompt

None.

Message

```
Block undefined or out of order
```

Begin and end markers must be set in the current window, begin before end, with the end marker to the right of the begin marker for vertical blocks.

See Also

Misc Undo, Delete Undo, and Window Undo for further information on restoring deleted text from the stack of delete buffers.

Environment Configure, Undo Buffers Max for further information on setting the maximum number of delete buffers that CoEdit will remember.

Restriction

Find All, Read Only

Description

Sets the block end marker at the current cursor location. If the block end marker had already been set, it is removed from its previous location.

Prompt

None.

Message

```
Line would be too long.
```

CoEdit has a maximum line length of 254 characters. If the current line is 254 characters long, no markers can be added to it.

Block ^BF

Filter

Description

Performs a user-definable filter operation on the defined block. (A filter is a program that reads information from standard input, changes it, and sends the resulting data out to standard output.) The currently defined block is then saved to the disk. That file is sent to the filter you specify and the standard output from the filter replaces the block in the window.

This is an extremely powerful command because it allows any operation to be performed on a block of text. For example, the operating system program sort could be used to sort a column of numbers or names. Any program can be written in your preferred language to perform any desired filter function. (See Appendix C for examples of filters.)

The block as it was before the filter was performed is yanked into the delete buffers for recovery. If you are satisfied with the result of the filter, remember to delete the block from the buffers (see Chapter 13 for the Internal Delete Buffers command), as the block may be of considerable size.

Prompt

```
Enter command:
```

Enter the filter name (for example, sort).

Message

```
Block undefined or out of order
```

Begin and end markers must be set in the current window, begin before end, with the end marker to the right of the begin marker for vertical blocks.

```
Filter Complete
```

The filter has performed its operation on the block of text.

See Also

Delete Undo for further information on recovering the old block from the delete buffers.

Restriction

Find All, Read Only

Block ^BL

Lower

Description

Converts all uppercase letters to lowercase within the current block. Only characters A through Z are modified.

Prompt

None.

Message

```
Block undefined or out of order
```

Begin and end markers must be set in the current window, begin before end, with the end marker to the right of the begin marker for vertical blocks.

See Also

Block Upper

Restriction

Read Only

Description

Moves the currently defined block in the current window to the cursor location. The block is deleted from its present location and inserted at the cursor.

Prompt

None.

Message

```
Block undefined or out of order
```

Begin and end markers must be set in the current window, begin before end, with the end marker to the right of the begin marker for vertical blocks.

```
Line would be too long
```

CoEdit has a maximum line length of 254 characters. If the block was copied to the current cursor location, a line longer than 254 characters would result.

See Also

Window Move for further information on moving text blocks between windows.

Restriction

Find All, Read Only

Description

Narrows the scope of the current window to include just the lines of the block. The Block Narrow process can be repeatedly nested; that is, you can narrow again within a narrow window.

Narrow windows are indicated by the nested depth enclosed within braces immediately after the file name on the window's border. You may not narrow a Find All window, but you may specify Find All in a narrow window.

The block as it was before the Block Narrow was performed is yanked into the delete buffers for recovery, but only if it has been changed. If you like the changes you made in the narrow window, remember to delete the block from the buffers (see Chapter 13 for the Internal Delete Buffers command), as the block may be of considerable size.

Prompt

None.

Message

```
Block undefined or out of order
```

Begin and end markers must be placed in the current window, begin before end, with the end marker to the right of the begin marker for vertical blocks.

See Also

Block Widen for further information on leaving Block Narrow mode.

Delete Undo for further information on recovering the old block from the delete buffers.

Window List for further information on how to see all of the open windows that are narrow.

Restriction

Find All

Description

Takes the standard output from any program or command and inserts that output into the window at the current cursor position.

Prompt

```
Enter command:
```

Enter the command or program name (for example, ls).

Message

```
Filter Complete
```

This message indicates successful completion of the filter.

Restriction

Find All, Read Only

Block

^BQ

Quick

Description

Sets the block markers around the current language statement. If the statement can be successfully parsed, and any block markers are already set, those block markers are removed.

The current statement is defined as the statement that begins on the line that the cursor is on, regardless of the cursor's position on the line. The current statement includes all statements within the statement itself. For example, if the current statement is an "if" statement, then the entire statement, including the "else" clause (if any) and any statements within the "if" or the "else" will be included in the block. If, as in Pascal, you block off a procedure declaration, the block will include all of the statements and nested procedures and functions within that procedure. Note that not all languages are supported.

Prompt

None.

Message

```
Syntax error in program
```

There is a syntax error in your program. For example, there may be a "do" loop in C with no "while" condition.

```
End of statement not found
```

The statement you have attempted to block off is incomplete. This may be an unclosed brace or a missing semicolon in C. Issue a Verify Syntax command to check your program syntax. (See Chapter 22.)

Restriction

Find All

Block ^BR

Remove

Description

Removes the begin and end markers from the current window if they have been set. They are completely reset, not just hidden. The cursor position is not changed.

Prompt

None.

Message

None.

Restriction

Find All

Description

Saves the text of the current block to a disk file.

Prompt

```
Save to which file?
```

Enter file name for the current block.

Caution

```
That file exists. Overwrite it?
```

The named file already exists. Press N to cancel Block Save. Note that this prompt will appear only if the environment option Safety is On. See Chapter 10 for details on the Safety option.

Block ^BT

Text Wrap

Description

Wraps and formats the text in the current block into a paragraph. The effect of this function is to "fill" the lines in the block with the text contained in the block in as few lines as possible, keeping the text within the margins. This is equivalent to reformatting a paragraph using a word processor. All white space at the beginning of lines is discarded and replaced with spaces to set the left margin. One space is placed between each word, except at the end of sentences, where two spaces are inserted.

The text will be right justified if the justification option is turned On with the Window Tabs (see Chapter 23) or the Environment Tabs commands (see Chapter 10). Spaces will then be added to the text so that it does not have a ragged right margin.

Upon completion of the command, the cursor is placed at the end of the block and the block markers are removed.

You can configure the left and right margins and the justification option for the current window with the Window Tabs command, and also set the defaults for all new windows with the Environment Tabs command. When the environment is saved to disk, these settings will be saved for future CoEdit sessions.

The old block (as it was before reformatting) is saved in the delete buffers so that this operation can be undone.

Prompt

None.

Message

```
Block undefined or out of order
```

Begin and end block markers must be set in the current window, begin before end, with the end marker to the right of the begin marker for vertical blocks.

See Also

Window Tab, Environment Tab, Delete Undo

Restriction

Find All, Read Only

Block ^BU

Upper

Description

Converts all lowercase letters in the current block to uppercase. Only characters a through z are modified.

Prompt

None.

Message

```
Block undefined or out of order
```

Begin and end markers must be set in the current window, begin before end, with the end marker to the right of the begin marker for vertical blocks.

See Also

Block Lower

Restriction

Read Only

Description

Toggles the block type from vertical to horizontal and back. This block type, which is displayed on the status line, determines the type of blocks that will be created via any command. The type of a block is also used to determine how it should be inserted into the text. (See the section "Inserting Block Text" earlier in this chapter.)

Prompt

None.

Message

None.

Block ^BW

Widen

Description

Takes a narrow window (one which is the result of a Block Narrow command) and replaces the parent window's block with its contents. If the block has been changed, the Block Narrow subwindow is inserted into its parent window in the original block position, and the original block is yanked into the delete buffers so that it can be restored.

Prompt

None.

Message

None.

Restriction

Find All, Not Block Narrow

Description

Copies the text of the currently defined block into the window's stack of delete buffers without deleting the text from the window. It can then be restored into another part of the file with the Misc Undo, Delete Undo, or Window Undo commands.

Prompt

None.

Message

```
Block undefined or out of order
```

Begin and end markers must be set in the current window, begin before end, with the end marker to the right of the begin marker for vertical blocks.

Chapter 8: Cursor Commands

Using Cursor Commands 8-1
Cursor Movement 8-1
    Movement On the Current Line 8-1
    Movement Between Lines 8-1
    Movement Within the Window 8-1
Cursor Bot Window 8-2
Cursor Down Line 8-3
Cursor End Window 8-4
Cursor Finish Line 8-5
Cursor Home 8-6
Cursor Left Word 8-7
Cursor Mid Window 8-8
Cursor Right Word 8-9
Cursor Start Line 8-10
Cursor Top Window 8-11
Cursor Up Line 8-12
Cursor End Word 8-13
Cursor Left One 8-14
Cursor Right One 8-15

Chapter 8: Cursor Commands

Using Cursor Commands

The Cursor commands allow you to move the cursor within the boundaries of the current window and to change the contents of the current window by scrolling new lines into it.

Note that while moving the cursor in the file, the cursor can never land "beyond" the ENTER character that signals the end of each line.

Cursor Movement

Cursor commands fall into three groups. There is movement on the current line, between lines, and within the window.

Movement On the Current Line

S start	F finish	
L word left	W word end	R word right
Y left one	Z right one	

Movement Between Lines

U up one D down one

Movement Within the Window

H upper left	E lower left	
T top line	M middle line	B bottom line

Cursor ^CB

Bot Window

Description

Moves the cursor to the last line in the window, maintaining the current columnar position if possible. If the new line is shorter than the line of the old position, the cursor moves to the end of the new line.

Prompt

None.

Message

None.

Description

Moves the cursor down one line, maintaining the current columnar position if possible. If the new line is shorter than the line of the old position, the cursor moves to the end of the new line.

If the maintain column position option is enabled (see Environment Configure command in Chapter 10), the cursor will either be moved towards the right to its previous columnar position or to the end of the new line, depending on which is closer to the left margin. This has no effect if the cursor is on the last line of the file. The window may scroll up if the cursor is on the last line of the window.

Prompt

None.

Message

None.

Cursor ^CE or End

End Window

Description

Moves the cursor to the last line in the window and sets the columnar position to one.

Prompt

None.

Message

None.

Description

Moves the cursor to the end of the current line.

Prompt

None.

Message

None.

Cursor ^CH or Home

Home

Description

Moves the cursor to the character at column one of the first line in the window. If the window is horizontally scrolled and the first column does not appear, the cursor moves to the end of that line and the window is scrolled towards the beginning of the line.

Prompt

None.

Message

None.

Description

Moves the cursor to the beginning of the previous word if the cursor is at the beginning of a word or on a delimiter. Moves the cursor to the beginning of the current word if the cursor is in the middle of a word. If the cursor has to move to the previous line, the window may scroll.

Prompt

None.

Message

None.

See Also

The Environment Configure command for information on setting the delimiters that determine word boundaries.

Cursor ^CM

Mid Window

Description

Moves the cursor to the line in the middle of the window, maintaining the current columnar position if possible. If the new line is shorter than the line of the old position, the cursor moves to the end of the new line.

Prompt

None.

Message

None.

Description

Moves the cursor to the beginning of the next word. If there are no more words on the current line, the cursor moves to the first word on next line; the window may scroll if the cursor is on the last line in the window.

Prompt

None.

Message

None.

See Also

The Environment Configure command, option P, for information on setting the delimiters that define a CoEdit word.

Cursor

^CS or Ctrl-Left Arrow

Start Line

Description

Moves the cursor to the start of the current line.

Prompt

None.

Message

None.

Description

Moves the cursor to the first line of the window, maintaining the current columnar position if possible. If the new line is shorter than the line of the old position, the cursor moves to the end of the new line.

Prompt

None.

Message

None.

Cursor

^CU or Up Arrow

Up Line

Description

Moves the cursor up one line, maintaining the current columnar position if possible. If the new line is shorter than the line of the old position, the cursor moves to the end of the new line.

If the maintain column position option is enabled (see Environment Configure command in Chapter 10), the cursor may be moved towards the right to its previous columnar position or to the end of the new line, whichever is closer to the left margin. This has no effect if the cursor is on the first line of the file. The window may scroll up if the cursor is on the last line of the window.

Prompt

None.

Message

None.

Description

Moves the cursor to the end of the current word if the cursor is in a word or to the end of the next word if the cursor is on a delimiter. (A word is a sequential set of characters ending in a blank space or a delimiter.)

If the cursor is at the end of the line, it moves to the end of the next word in the file.

Prompt

None.

Message

None.

See Also

The Environment Configure command for information on changing the delimiters that define a word in CoEdit.

Cursor ^CY or Left Arrow

Left One

Description

Moves the cursor left one position. If the cursor is at the beginning of a line, then it is placed at the end of the previous line. If it is at the beginning of the file, then no action is taken.

Prompt

None.

Message

None.

Description

Moves the cursor right one position. If the cursor is at the end of a line, then it is placed at the beginning of the next line. If it is at the end of the file, then no action is taken.

Prompt

None.

Message

None.

Chapter 9: Delete Commands

Using Delete Commands 9-1
Delete Command Summary 9-1
Delete To Bottom of File 9-3
Delete To Character 9-4
Delete Line 9-5
Delete To End of Screen 9-6
Delete To Finish of Line 9-7
Delete To Home 9-8
Delete Word Left 9-9
Delete One 9-10
Delete Prev 9-12
Delete Word Right 9-13
Delete To Start of Line 9-14
Delete To Top of File 9-15
Delete Undo 9-16
Delete Word 9-18
Delete Yank 9-19

Chapter 9: Delete Commands

Using Delete Commands

The Delete commands remove text from the current window. The range of the text to be removed is always determined with reference to the current cursor location. For example, Delete to Start removes all text from the cursor to the start of the current line.

Text deleted by CoEdit is not completely discarded. The text removed by a Delete command is added to a Last-in-First-Out (LIFO) stack of delete buffers. The maximum size of this stack and whether or not single characters are buffered can be set with the Environment control command (see Chapter 10). As more buffers are added to the stack, the oldest buffers are removed when the stack is at its size limit.

The Misc Undo command will undo the last deletion by inserting the first entry on the delete buffer stack into the text at the current cursor location. The Delete Undo command displays the delete buffer stack and prompts you for the number of the buffer to insert back into the text.

The Delete Yank command is identical to the Miscellaneous Undo command except that it does not remove the last deletion from the buffer stack; thefore, the Delete Yank command allows multiple copies of a buffer to be inserted into the file. Note that each window has an independent buffer stack; it is possible to delete a line in one window, move to another window, and undo that line in the first window with Window Undo (see Chapter 23 concerning Window commands). See the Window Copy and Move commands for inter-window text transfer.

Delete Command Summary

The Delete commands can be arranged as follows:

From the cursor backward:

TOF	top of file	^DT
Home	home of window	^DH

Start	start of line	^DS
Left Word	begin of word	^DL

From the cursor forward:

Right Word	end of word	^DR
Finish	end of line	^DF
End	end of window	^DE
BOF	bottom of file	^DB

Others:

To a char	to a character	^DC
Line	the current line	^DD
Current	current char	^DO/DELETE
Prev Char	previous char	^DP/BKSP
Undo	undo any delete buffer	^DU
Word	the current word	^DW
Yank	yank delete buffer #1	^DY

Related Commands

See also Chapter 25 for descriptions of Yank commands. The Yank commands mirror the Delete commands exactly except that the Yank commands do not actually remove the text. Yank commands copy text into the buffers; that is, they "yank" it into the buffer stack.

See also Chapter 7 for further information on inserting text.

Description

Deletes all text from the current character to the last character in the file.

Prompt

None.

Message

None.

Restriction

Find All, Read Only

Delete ^DC

To Character

Description

Deletes all text from the cursor to a character you specify. The character specified must be on the current line or the command has no effect.

Prompt

```
To what character?
```

Enter the character immediately following text that should be deleted.

Message

```
Character not found
```

The character you entered could not be found after the cursor on the current line.

Restriction

Read Only

Description

Deletes the entire current line. The cursor moves to the next line unless the current line is the last in the file; then, it moves to the previous line. The cursor will move to the left margin (the beginning of the line). The window contents will scroll as necessary to keep the last line of the file on the bottom of the window or the first line of the file on the top of the window.

Prompt

None.

Message

None.

Restriction

Find All, Read Only

Delete ^DE

To End of Screen

Description

Deletes all characters from the cursor to the last character shown in the current window.

Prompt

None.

Message

None.

Restriction

Find All, Read Only

Description

Deletes all characters from the cursor to the end of the current line. The command has no effect when the cursor is at the end of the line.

Prompt

None.

Message

None.

Restriction

Read Only

Delete ^DH

To Home

Description

Deletes all characters from the cursor to the first character shown in the current window.

Prompt

None.

Message

None.

Restriction

Find All, Read Only

Description

Deletes from the cursor to the beginning of the current word, if the cursor is in the middle of a word.

Deletes all delimiters to the end of the previous word (or the beginning of the line) if the cursor is at the beginning of a word or on a delimiter.

The command has no effect at the beginning of the line.

Prompt

None.

Message

None.

Restriction

Read Only

See Also

Environment Configure for information on delimiters and how they affect this command.

Delete

^DO or DELETE

One

Description

Deletes the character at the cursor and shifts to the left any characters that are to the right of the cursor. If the current line contains only an ENTER character, the command is equivalent to the Delete Line command. If the cursor is at the end of the line, the ENTER character is deleted and the next line is concatenated onto the current line.

Prompt

None.

Message

```
Line would be too long
```

CoEdit has a maximum line length of 254 characters. If the character being deleted is the ENTER character, then the result of concatenating the two lines would exceed 254 characters. The command is ignored.

Restriction

Read Only

See Also

Environment Configure for information on controlling whether or not CoEdit will remember deleted characters.

Delete ^DP or BACKSPACE

Prev

Description

Deletes the character located just before the cursor and shifts to the left any characters that are the right of the cursor. If the cursor is at the beginning of the line, the ENTER character is deleted and the current line is concatenated onto the previous line.

Message

```
Line would be too long
```

CoEdit has a maximum line length of 254 characters. If the character being deleted is the ENTER character, then the result of concatenating the two lines would exceed 254 characters, and the command is ignored.

Restriction

Read Only

See Also

Environment Configure for information on controlling whether or not CoEdit will remember deleted characters.

Description

Deletes the entire word if the cursor is at the beginning of a word.

Deletes from the cursor to the end of the current word if the cursor is in the middle of a word.

Deletes all delimiters to the beginning of the next word (or end of line) if the cursor is on a delimiter.

The command has no effect at the end of a line.

Prompt

None.

Message

None.

Restriction

Read Only

See Also

Environment Configure for information on delimiters and how they affect this command.

Delete ^DS

To Start of Line

Description

Deletes all characters from the left of the cursor to the start of the line. The command has no effect when the cursor is at the start of the line.

Prompt

None.

Message

None.

Restriction

Read Only

Description

Deletes all characters from the left of the cursor to the first character of the file. The character at the cursor then becomes the first character of the file.

Prompt

None.

Message

None.

Restriction

Find All, Read Only

Delete ^DU

Undo

Description

Displays the stack of delete buffers and prompts for the number of the delete buffer. Delete Undo will then insert the text from the delete buffer specified into the text at the cursor location. Note that the text is not inserted at the location from which it was deleted; it is inserted at the current cursor position.

Prompt

```
Deletion to undo
```

Enter the number that corresponds to the buffer you want to insert. The buffers are listed by number with the number of lines, the type (vertical or horizontal), and the beginning and ending text displayed. If there is only one delete buffer, it will be the default input.

Message

```
Line would be too long
```

CoEdit has a maximum line length of 254 characters. Inserting the chosen buffer would create a line longer than this maximum; therefore, the command is canceled.

See Also

"Inserting blocks of text" in Chapter 7 for information on how text is inserted into the file.

See also the Misc Undo command, which performs a Delete Undo on the first delete buffer without prompting, and the Window Undo command, which performs a Delete Undo for deleted text from other windows into the current window.

Restriction

Read Only

Delete ^DW

Word

Description

Deletes the entire current word if the cursor is on a non-delimiter. Deletes all delimiters forward to the next word (or end of line) if the cursor is on a delimiter. If the cursor is at the end of a line, the RETURN character is deleted, thereby appending the next line onto the current line.

Prompt

None.

Message

```
Line would be too long.
```

CoEdit has a maximum line length of 254 characters. The line might be too long if the current line is very long and the cursor is at the end of the line.

Restriction

Read Only

See Also

Environment Configure for information on delimiters and how they affect this command.

Description

Yanks (copies) the first delete buffer to the current cursor position, leaving the buffer intact on the delete stack. In this way you may copy the buffer into your text as many times as you like.

Prompt

None.

Message

None.

Restriction

Find All, Read Only

See Also

Misc Undo for information on undoing the first delete buffer.

Chapter 10: Environment Commands

Using Environment Commands 10-1
Saving the CoEdit Environment 10-1
co.cfg 10-2
co.nfy 10-2
Menu Definition 10-2
Menu Bindings 10-3
Function Key Bindings 10-3
Environment Attribute 10-5
Environment Configure 10-6
A. Autoindent 10-6
B. Horizontal Scroll 10-6
C. Command Mode 10-7
D. Safety Prompts 10-7
E. Pad ASCII Files 10-7
F. Cursor Top Line 10-8
G. Cursor Bottom Line 10-8
H. Help Menus 10-8
I. Insert Mode 10-9
J. Beep Tone 10-9
K. Duration 10-9
L. Pitch 10-10
M. Move on Yank Line 10-10
N. Find File 10-10
O. Error Directory 10-10
P. Delimiters 10-11
Q. Loc/Rep Ignore Case 10-11
R. Backup Levels 10-11
S. Showmatch 10-12
T. Buffer Characters 10-12
U. Undo Buffers Max 10-12
V. Evaluator Base 10-12
W. New Windows 10-13
X. Expand New Windows 10-13
Y. Yes/No Return 10-13
Z. Clear Screen 10-13
1. Maintain Column Pos 10-14
2. Prototyping 10-14
3. Autosave Mode 10-14
4. Autosave Period 10-14

5. Show Side Borders 10-15
6. Show Line Continuation 10-15
Environment Language 10-17
A. Compiler 10-17
B. Error Extension 10-18
C. Command 10-18
Environment Macro 10-20
A. Macro Extension 10-20
B. Macro Library 10-20
C. Automatic Macro 10-20
D. Maximum macro size 10-21
E. Begin Key Scan Code 10-21
F. End Scan Key Scan Code 10-22
G. Default to Library 10-22
Environment Printer 10-24
Environment Read 10-26
Environment Save 10-27
Environment Tab 10-28
Display 10-28
Type 10-28
Width 10-29
Stops 10-29
Justification 10-29
Left Margin 10-30
Right Margin 10-30
Environment Update 10-32

Chapter 10: Environment Commands

Using Environment Commands

The environment is the complete set of operating parameters that affect the way in which CoEdit performs. You can define many of these parameters, including the key definitions, the screen attributes, and miscellaneous environment variables.

The Environment command group contains three disk commands: one to read the environment from the disk and two to save it to the disk. The other environment commands change the configuration of the program. All parameters that can be defined with the Environment commands, as well as key bindings, will be saved with the Environment Save command.

Saving the CoEdit Environment

The environment is saved from execution to execution of CoEdit in two disk files, co.cfg and co.nfy, known as configuration files.

CoEdit looks for the configuration files in the current directory when it loads. If neither configuration file can be found, then the master directory is searched. If CoEdit finds co.cfg but not co.nfy in the current directory, then the master directory will not be searched for co.nfy, and CoEdit will use the default key bindings. (See Chapter 5 for information about the master directory.) If no co.cfg can be found in the current or in the master directory, the default environment is used, with all options and key bindings set to their default values.

Several different versions of the environment can be stored on the disk in different directories. The environment is read from a configuration file if you change directories by executing the System Goto command or by executing a command that changes the current directory.

As soon as the configuration file is read, the new environment is in effect. This allows different environment parameters to be set for different projects residing in different directories, as well as for different users in different directories. Since the configuration file is in the master directory, a global configuration may be used from every directory and by every user without having multiple copies of the files. When CoEdit changes directories during program execution, the new directory is searched. If no configuration file is found in the new directory, then the master directory will not be searched, and the current configuration will be left unaltered.

co.cfg

co.cfg contains all of the CoEdit environment except the key bindings. Caution: You should not try to modify the co.cfg file.

co.nfy

co.nfy is an ASCII file that contains CoEdit's menu definitions and key bindings. You may use CoEdit to edit this file. The file is divided into three sections. The beginning of each section is indicated by a line beginning with two number signs (##), which may be optionally followed by a comment label.

Menu Definition

The first section is the Menu Definition section. It tells CoEdit whether any given menu is accessed by a Control- or an Alt- key, the name of the menu, and the title of the menu, as illustrated in the following two examples:

```
Menu CtrlK "WORDSTAR BLOCK" "Block"

Menu AltW "WINDOW" "Wind"
```

The first of the preceding examples creates a menu bound to Ctrl-K. When you press Ctrl-K, the word Block appears in the message area at the upper left of the screen. When the menu for Ctrl-K appears, WORDSTAR BLOCK MENU will be displayed at the top of the window. In this example, the name and title of the menus must be enclosed in double quotes as shown. The four elements on the line ("Menu", "CtrlK",

the title, and the name) must be separated from each other by at least one space or tab.

The second of the preceding examples defines the Window menu as an Alt menu, meaning that this menu is invoked by pressing Alt-W.

Menu Bindings

The second section is the Menu Bindings section. It defines which CoEdit commands are bound to which menus. The format of these lines is simply keystroke=command.

Here are some examples:

```
WP=Window_previous
TF=Jump_to_bof
```

The first example binds the command Window_previous (Window Previous) to the key sequence ^WP (or Alt-WP if in the Menu Definition section you told CoEdit that the Window menu was an "Alt menu", as we did in our examples above). WP is the default for CoEdit.

The second example binds the Jump_to_bof (Jump to Top File) command to the key sequence ^TF (for Top File).

Note that the menu group specified on the left side of the equal sign (W, K, and T in our examples) must have been defined in the Menu Definition section; otherwise, they will be ignored. There may be no white space in the definitions. In other words, each line must have two letters, an equal sign, and the command name with no spaces between them.

Function Key Bindings

The third section is the Function Key Bindings section. All other key bindings not specifically referring to menus are defined here. For example, you can define F1, AltF10, Shift7 (Home), Ins, and virtually any other non-ASCII key listed in Appendix F. The format of the bindings in this section is the same as for those in the Menu Bindings section. Here are some examples:

```
CtrlEnd=Go_end_line
Ins=Undo
Enter=Auto_return
```

The first example binds the Go_end_line (Cursor Finish line) command to the keystroke CtrlEnd. The second binds the Undo (Undo) command to the Ins key. The last example makes the Enter key execute the Auto_return command, which performs the usual Enter key function, followed by an autoindentation. (This is effectively the same as turning on Autoindent. See Environment Configure later in this chapter.)

Note that, as in the Menu Bindings section, there may be no white space in the lines, before or after the key, equal sign, or command.

There is one important variation on the two key binding sections in the co.nfy file. The command that is bound to a keystroke need not be a CoEdit command. It can be a CoEdit macro. This allows any macro you create to be a direct extension to the set of CoEdit commands. To do this, the command (on the right side of the equal sign) is simply replaced by an asterisk (*) followed by the name of the macro. Examples are:

```
KD=*FILEDONE
Enter=*MYENTER
```

The first binds the macro FILEDONE to the key sequence ^KD. The command will appear on the K menu (however that may be defined). The short description next to the highlighted D will be the name of the macro. The long description displayed at the bottom of the menu will be the description of the macro. Thus the FILEDONE macro will be indistinguishable from the regular set of CoEdit commands.

We recommend that you set up a key binding that assigns the CoEdit temporary macro (see Chapters 5, 6, and 17) to a key or key sequence.

The second example simply binds the macro MYENTER to the Enter key.

Here again, there may be no white space in the lines, before or after the key, equal sign, or macro name.

The remainder of this chapter presents and describes the individual environment commands.

Description

Sets the screen character attributes. This command sets the background and foreground color, intensity, and blinking features of various sections of the screen.

Enter 'C' for color or 'M' for monochrome at the prompt. The screen will then fill with the names of the attributes that can be changed and sample texts displayed with the current attribute value. Move the highlighted bar to the attribute name to be changed and press ENTER. A list of the available screen attributes will appear. Use the arrow keys to move the cursor to the desired attributes. The sample text will change to reflect the selected attribute. When you are satisfied, press ENTER. To cancel selection for this named attribute, press ABORT. When you have completed your changes, press ABORT to complete the command.

Prompt

```
Color or mono?
```

Enter 'C' to set the color attributes, 'M' for the monochrome.

```
Enter selection:
```

Move the highlighted bar to the named attribute to change.

Description

Sets general CoEdit configuration variables. At the prompt, select a letter corresponding to the option you want to select. If the selection is an On/Off toggle, then the new value is displayed after the option description and you are free to make another selection. If the selection requires either numeric or string data, the selection will be highlighted and the cursor placed at the input area. Enter the new information. The usual cursor keys are available.

The following is a complete description of the options that may be configured.

A. Autoindent

The autoindent feature tells CoEdit to indent the cursor to the position of the first non-blank character on the previous line when the ENTER key is depressed and Insert mode is On. The spaces and tabs at the beginning of the line above are duplicated exactly.

Default Value: Off

B. Horizontal Scroll

The value for horizontal scroll tells CoEdit how many columns to shift the screen left when the cursor moves right beyond the limit of the screen. Small values provide smooth scrolling with constant movement; large values provide jerky but less frequent movements. The maximum value is 50.

Default Value: 15

C. Command Mode

Toggles the Command Mode in CoEdit. See Chapter 4 for a description of Command Mode.

Default Value: Off

D. Safety Prompts

Safety Prompts tell CoEdit whether to ask precautionary questions and request confirmations when an action with potentially serious consequences may occur. For example, if Safety Prompts are On and you issue a Quit Terminate command, CoEdit will see if any windows have been modified. If any have been, CoEdit will ask you for confirmation.

Default Value: On

E. Pad ASCII Files

Indicates to CoEdit whether it should pad ASCII files to the next 128-byte sector with Control-Zs, as some word processor and database programs require.

Default Value: Off

F. Cursor Top Line

Specifies the top scan line for the cursor, thereby allowing the user to change the size of the cursor. On a monochrome screen, the minimum number is 0; the maximum is 13. On a color screen the minimum is 0; the maximum number is 7. See also the following option G.

Default Value: 0

G. Cursor Bottom Line

Specifies the bottom scan line for the cursor, thereby allowing the user to change the size of the cursor. The minimum number is 0; the maximum is 13. On a color screen the minimum is 0; the maximum number is 7. See also the previous option F.

Default Value: 13 for monochrome, 7 for color

H. Help Menus

This specifies whether or not CoEdit should display the help menus. Help menus normally appear a few seconds after a command group letter has been struck. Note that if help menus are turned off, then pressing a "?" after pressing the command group letter will force CoEdit to display the menu for that command group.

Default Value: On

I. Insert Mode

This toggles the insert mode. See Chapter 4 for a description of Insert Mode. See also the Goto Insert and Goto Overwrite commands in Chapter 15.

Default Value: On

J. Beep Tone

This turns On or Off the warning beeps in CoEdit.

Default Value: On

K. Duration

This sets the length of the warning tone that CoEdit generates.

Default Value: 200

L. Pitch

This sets the pitch of the warning tone that CoEdit generates.

Default Value: 150

M. Move on Yank Line

This tells CoEdit whether it should move the cursor down one line when the user issues a Yank Line command (see Chapter 25). Turning this option On allows the user to easily copy a series of lines to another part of the window.

Default Value: On

N. Find File

Turning this option On tells CoEdit to find the named file when it asks for a filename. See Chapter 4 for a description of how CoEdit looks for files.

Default Value: On

O. Error Directory

This directory is the one in which CoEdit will look for error files generated by the compilers. See the Environment Language command later in this chapter for more information.

Default Value: the current directory at program load time

P. Delimiters

This option sets the delimiters in CoEdit. The delimiters will determine the way in which many of CoEdit's commands will work, particularly the Cursor, Delete, and Yank commands. Experimentation will be your best guide to setting the right delimiters for your work. The tilde character (~) may be used to denote a tab character.

Default Value: > [ ] (; : , . { } ~

Q. Loc/Rep Ignore Case

Turning this option On will cause CoEdit to ignore the case (that is, upper or lower) of search strings by default. Thus, LPI would match lpi. If this option is Off, they would not match. This option is ignored when using the UNIX version's UNIX-compatible regular expressions.

Default Value: On

R. Backup Levels

This tells CoEdit how many backup files you want it to create for each original file you edit. The maximum is 36.

Default Value: 1

S. Showmatch

If this option is On, CoEdit will temporarily move the cursor to the corresponding opening match if any one of the characters),], or } is typed. If no match is found, CoEdit will beep. If the next character is struck before CoEdit can find the match, the search is canceled.

Default Value: On

T. Buffer Characters

If this option is On, CoEdit will not save single deleted characters. This is useful for not cluttering up the deletion stack.

Default Value: Off

U. Undo Buffers Max

This number is the size of the deletion garbage stack. In other words, this is the delete buffer stack. The maximum is 32765; the minimum is 1.

Default Value: 15

V. Evaluator Base

This number is the default base for the expression evaluator (see the description of Miscellaneous Evaluate in Chapter 26.) Valid bases are 2 (binary), 8 (octal), 10 (decimal), and 16 (hexadecimal).

Default Value: 10

W. New Windows

This determines whether or not new windows will be split vertically or horizontally. Pressing W will toggle this option.

Default Value: Split H.

X. Expand New Windows

If this option is On, CoEdit will open all new windows with the expanded attribute. This means that all new windows will occupy the entire screen.

Default Value: Off

Y. Yes/No Return

When this option is On, CoEdit will require an ENTER keystroke to confirm the response on prompts that require a Yes/No input. If it is Off, CoEdit will accept the first Y or N it reads from the keyboard.

Default Value: Off

Z. Clear Screen

If this option is On, CoEdit will clear the entire screen when it is terminated and is returning to the operating system.

1. Maintain Column Pos

If this option is On, CoEdit will maintain the cursor's columnar position regardless of current line length. Any editing command (such as typing, deleting, or undoing) will reset the maintained position to the current cursor position.

Default Value: Off

2. Prototyping

If this option is On, CoEdit will enable or disable keyword prototyping.

Default Value: Off

3. Autosave Mode

This option is set either to Off, Keystroke, or Minutes. If it is Off, autosaving is disabled. If it is set otherwise, the value in the following option determines the period between automatic savings. See Chapter 1 for more information on automatic file saving.

Default Value: Off

4. Autosave Period

This is a numeric field that determines the period between automatic file savings. For example, if this value is 30, then automatic file saving will occur every 30 keystrokes or every 30 minutes, depending on the value of option 3, above. The maximum value is 32767.

Default Value: 0 (Off)

5. Show Side Borders

When this option is On, the left and right borders will be displayed for the windows. Otherwise they will not be displayed, adding two columns of width for editing.

Default Value: On

6. Show Line Continuation

If this option is On, the left and right borders will display a "tick" mark to indicate that the text line extends beyond the window border.

Default Value: Off

Prompt

```
Enter selection:
```

Enter a letter or number corresponding to the option you want to select.

Message

```
Number too high -- range is XXXX to YYYY
```

The number you have entered is too large. Enter a number within the range.

`Number too low -- range is` *XXXX* `to` *YYYY*

The number you have entered is too small. Enter a number within the range.

`Bad base - 2, 8, 10, or 16`

The evaluator base you have entered is invalid. Enter a valid choice.

See Also

Locate String for information on the tilde (~) tab character.

Description

Sets programming language information for CoEdit. This command does not set a "current language" for CoEdit; rather, it defines the information CoEdit needs when that language is in use. When a language is in use it is defined by the extension of the file in the current window. See the "Verify Commands" section in Chapter 22 for further information. With this option, the compiler, the name of the error file, and the command used to invoke the compiler can be specified. At the select a language prompt, enter a letter corresponding to the language you want to select.

A. Compiler

This option defines the compiler that you are using for the given language. Press A until the compiler you use appears.

Values:

LPI-BASIC
C System compiler
LPI-C
LPI-COBOL
LPI-FORTRAN
LPI-PASCAL
LPI-PL/I
Iliad Group for CML

B. Error Extension

This tells CoEdit the extension of the error files generated by your compiler or compilation batch file (see option C, below). Generally, it is ERR. Note: Do not enter the period/dot (.) here.

Default Value: None

C. Command

This is the command CoEdit will use to invoke your compiler. It should probably be a shell script. a batch file or a shell script. CoEdit will look for the special symbols in the compiler command which will be substituted by an item. The symbols are:

SYMBOL	ITEM
%p	the full path name of the file without the file's extension
%d	the directory of the file
%n or %s	the file name without extension
%e	the file entension without the '.'

Every occurrence of one of these symbols will be replaced with the corresponding item. If CoEdit cannot find any of these symbols, it will simply append the file name (%s) to the end of the command. For example, if the command were CC, and the file named MYPROG.C, the resulting command would be CC MYPROG. If the command were masm %s,,%s;, and the file name was VIDEO.ASM, the resulting command would be masm VIDEO,,VIDEO;.

See the Verify Compile command in Chapter 22 for examples of batch files you might use to invoke your compiler.

Default Value: None

Prompt

```
Select a language:
```

Enter the letter corresponding to the language you want to select.

```
Select an item:
```

Enter an A, B, or C to select the compiler code, error extension or compiler command.

Message

None.

See Also

Verify Commands for information on using these variables.

See also Environment Configure; use the Error directory option to set the error output directory.

Description

Sets CoEdit macro information. There are seven pieces of information needed to configure the way macros work in CoEdit.

A. Macro Extension

This is the extension of the macro files on the disk. CoEdit will look for disk files with this extension to identify macros. If this extension is changed, the current macros will not be affected. Note: Put in the dot (.) as the first character here.

Default Value: .cmf

B. Macro Library

This is the directory in which CoEdit will look for macro files. The directory that is current when CoEdit is loaded will also be searched. If this is changed, then the Macro Set command described in Chapter 17 can be used to make CoEdit recognize macro files in the new directory.

C. Automatic Macro

This is the macro that CoEdit will execute each time CoEdit is loaded. It may be used for stream editing or for initial setup of CoEdit. See Appendix D for more information regarding stream editing.

Default Value: the current directory at program load time.

D. Maximum macro size

This is the maximum size macro that CoEdit will retain in memory. Macros are loaded into memory when they are invoked. This command sets the maximum size of a macro. Storing a macro in memory allows an immediate response to subsequent macro invocation. If this number is set to zero, all macros will be read and interpreted from their disk files each time they are invoked. See also the Macro Flush and Macro Set commands in Chapter 17.

Default Value: 512

E. Begin Key Scan Code

Sets the scan code for the Macro Definition key. The scan code must be entered in hexadecimal (base 16). (For keyboards without scan codes, CoEdit maintains an internal scan code for each key.) CoEdit's defaults are F9 (scan code hex 43) for Macro Define and F10 for Macro End Definition (see scan code hex 44 in the following section). This option allows you to determine which key activates this function. (For a description of Macro Define and Macro End Definition, see Chapter 17. Be careful, though. The scan code of hex 32 means not only M, but also Shift-M, Control-M, and Alt-M. See also the Internal Key Code command in Chapter 13 for a way to identify scan codes for all keys on the keyboard while you are editing.

Default Value: Hex 43 (F9)

F. End Scan Key Scan Code

Sets the scan code for the Macro End Definition key. See the notes for the previous Option.

Default Value: Hex 44 (F10)

G. Default to Library

Option G indicates whether newly added macros will be stored in the current directory or in the defined macro library (see option B). This feature is very useful, but it must be used with caution. For example, suppose you set this option to On, so that CoEdit stores new macros in the macro library. Now you define a macro and bind it to F1.

The macro disk file is stored in the macro library (presumably not the current working directory). If there is a macro for F1 in the current working directory, and you strike F1, then the macro from the current working directory is invoked, NOT the one you just defined and stored in the macro library. This is because macros in the current directory take priority over the ones in the library.

Default Value: Off

Prompt

```
Enter a letter to select an item
```

Enter a letter from A through G to select an option.

Message

None.

See Also

Chapter 5 for information on macros in general, and Chapter 17 for a complete description of all macro commands.

Environment ^EP

Printer

Description

Allows you to define the commands CoEdit needs to interact with your system's line printer command (usually lp or lpr). For a complete description of lp, please consult your UNIX manual.

The information you need to supply is:

A. The command to print a file. This is the command CoEdit will use to output information to the printer. Although usually lp, this command may also be a script that invokes a filter program to format the text before printing and then prints the file. When entering the print and eject (see next item) commands, use the following symbols for substitution:

SYMBOL	ITEM
%c	the number of copies to be printed
%d	the destination printer name
%n or %s	the complete file name
%o	optional command line argument (positional indicator only)

B. The command to eject the printer one or more pages. If this command is blank, CoEdit will do nothing with Print Eject requests.

C. The default printer name. For lp, this is the name of the system default printer (for example, lpr, or lineprinter).

D. The command line switch (for the command you specified in option A) to make that command delete the file after it is done printing. For lp, this switch is usually -r. When required, CoEdit will use this switch in the position determined by the %o in the command line specified in option A, above.

The typical default values are as follows:

A. lp %o -s -n%c -d%d %s

B. echo ^L | lp

C. lpr (or whatever your system's printer is)

D. -r

Prompt

```
Enter a letter to select an item:
```

Enter 'A' through 'D' to change the printer interface commands.

Message

None.

Environment ^ER

Read

Description

Reads configuration information from a directory you specify. The default directory is the current directory. The environment information will be immediately updated and in effect.

Prompt

```
Read from which directory?
```

Press ENTER to accept the default response or enter a new response and then press ENTER.

Message

```
No such directory
```

The directory you have entered does not exist.

```
No configuration file found
```

The directory you specified contains no configuration file.

Description

Saves the current environment on the disk. CoEdit will ask you for the directory where the configuration file and the key file should be written. The default response will be the directory from which the configuration files were last read. If no configuration file has been read, the default will be the current directory on the current disk. Configuration files existing in the directory you specify will be overwritten without warning.

Prompt

```
Save to which directory?
```

Press ENTER to accept the default response or enter a new response and then press ENTER.

Message

```
No such directory
```

The directory you have entered does not exist.

Environment ^ET

Tab

Description

Sets default tab configuration. Setting the tabs with the Environment Tab command does not affect any open windows. Only windows opened subsequent to the Environment Tab command will have the new configuration. The Environment Save command will save the tab configuration as defined here, not as defined in the current window. See Chapter 21 for a discussion of tabs in CoEdit. To change tabs in the current window, see the Window Tabs command in Chapter 23.

Display

This toggles the display of the otherwise invisible soft tab character so that it becomes visible as a small character resembling two less-than signs (<<). If other tabbing methods are in effect (see the following section), this option has no effect.

Default Value: Off

Type

This selects the type of tabs you want in your new windows. There are three types: soft, hard, and stops. Soft uses the soft tab character (ASCII 9 or ^I). This allows the easy maintenance of columns. Hard tabs are simply spaces, but are similar to soft tabs in that they are set at a preset width (see option 'W', below). The last, fixed tab stops, are tabs that you can set in any column from 1 to 255, much like a standard typewriter.

Default Value: Soft

Width

This allows you to set the width of the soft and hard tabs in new windows. The width may be from 1 to 16. This value is ignored when using fixed tab stops.

Default Value: 8

Stops

This allows you to set fixed tab stops. When this option is selected, the cursor will appear on the first line of the ruling. The cursor may then be moved around by using the Up, Down, Left, Right, Home, End, CtrlLeft, and CtrlRight keys. These keys have the same effect that they would have in an editing window. The column in which the cursor is located will be displayed on the left side of the ruling. To set a fixed tab, press the Ins key. A tab indicator will appear there. To remove a tab, place the cursor on a tab and press Del. The selected stops will appear below the ruling. A maximum of 20 fixed tab stops are permitted.

Default Value: No stops defined

Justification

This allows you to have the Block Text Wrap command right-justify text. On means right justification is enabled, Off means that it is disabled.

Default Value: Off

Left Margin

Sets the left margin for the Block Text Wrap command. This is the columnar position to which this text should be indented on the left.

Default Value: 9

Right Margin

Sets the right margin for the Block Text Wrap command. This is the columnar position after which no text should appear.

Default Value: 72

Prompt

```
Enter a letter to select an item:
```

Enter a D, T, W, S, J, L, or R to select an item. Press ENTER to accept the changes, or either Esc or ^C to abort the changes.

Message

```
Number too low - range is 1 to 16
```

The number you entered for the tab width is too small. Select a number in the given range.

```
Number too high - range is 1 to 16
```

The number you entered for the tab width is too large. Select a number in the given range.

```
Right margin must be greater then left margin
```

The right margin must be beyond the left margin for proper word processing.

See Also

Window Tab for information on setting tabs for the current window only.

Environment ^EU

Update

Description

Updates the configuration file and the key file on the disk. These configuration files will be written to the directory from which they were last read. If no configuration has been read from the disk, the environment will be written to the current directory on the current disk.

Prompt

None.

Message

None.

Chapter 11: File Commands

Using File Commands 11-1
File Names 11-1
^F processing at prompts 11-1
    Automatic File Finding 11-2
    Control-L Processing at File Prompts 11-2
    Special Multiple File Processing 11-2
File Another 11-4
File Done 11-5
File Exchange 11-7
File Load 11-9
File Merge 11-11
File Next 11-12
File Open 11-14
File Read 11-15
File Save 11-16
File Update 11-17

Chapter 11: File Commands

Using File Commands

The File commands perform almost all movement of text between CoEdit windows and the disk (see also Block Write and Quit Save-Exit). Files can be saved to the disk, loaded from the disk into the current or a new window, and merged from the disk into the current window.

The most basic commands are the following:

- File Load (^FL), which reads a disk file into the current window
- File Open (^FO), which reads a file into a new window
- File Done (^FD), which saves the file back onto the disk and closes the window

File Names

All File commands accept full operating system path names (see your operating system manual for details on naming files).

^F processing at prompts

Almost all File commands will require user input of one or more file names. CoEdit has a very simple and powerful way to let you scan disk directories while at a File command prompt. This feature allows you to display on the screen all or some of the files that reside in a given directory without interrupting the command being executed. At a file command prompt, pressing ^F will cause CoEdit to take the current input field as the directory and file pattern to display. For example, typing the following:

```
Save to which file? /sys/msc/*.dta^F
```

will display all .dta files in the msc subdirectory of directory sys. You may change the text in the input field and press ^F again to view a

different directory or pattern. If there is no file pattern, CoEdit displays the entire directory; if the input field is empty or there is no directory, the entire current directory is displayed. If an illegal directory is entered, no files are displayed.

The ^F feature will display as many matching files as possible on the screen. Use the PgUp and PgDn keys on the numeric keypad to view the files. If your terminal does not have a keypad, use the F2 (PgUp) and F3 (PgDn) keys. Each file is displayed with the date and time of its last update and its size in bytes.

When file names are displayed using ^F, you may use the Up and Down arrow keys to move the highlighted bar over each name. Pressing ^S will select the current highlighted file and copy it to the input area.

Automatic File Finding

If the environment variable Find File is On, then CoEdit will automatically locate disk files when reading files. All you have to do is enter the basic name of the file. See Chapters 1 and 10 for a complete description of the Find File feature.

Control-L Processing at File Prompts

Pressing ^L (Ctrl-L) when entering a file name will ask CoEdit to look for the file you have entered. If it can be found, CoEdit will replace the name you have entered with the full path name of the file it found. If it could not find a match, the name you entered remains unchanged.

```
Read from which file? sg.c^L
```

If the file SG. was found, CoEdit will display something like this:

```
Read from which file? /sg/sg.c
```

Special Multiple File Processing

It is possible to enter a file pattern at the prompts for the File Open and File Read commands. With CoEdit's multiple file processing system, you will be able to read all of the files matched by the pattern, either one at a time or all at once.

The File Read command will open all files selected by your pattern and load them into new windows. As many windows are created for these files as CoEdit has memory. If all the files cannot be loaded at once, CoEdit remembers the next file to be loaded and ends the command. Those files may be loaded using the File Next or File Another commands.

The File Open command will open only the first file selected by your pattern and load it into a new window. CoEdit remembers the next file to be loaded and ends the command. The rest of the files may be loaded with the File Next or File Another commands.

Both File Open and File Read can be used to create a new window and read only a single, non-wildcard file such as MYPROG.ASM.

The File Another and File Next commands may be used to continue processing multiple files. File Another opens a new window and loads the next file into it. File Next saves the current window if it has been modified and loads the next file into this window. In this way, you can sequentially edit all files that match a file pattern without knowing the names of the files.

Using multiple file processing and CoEdit's CML macro language, you can create powerful macros to alter an entire set of files automatically. (See Chapter 6 for information on the Morefiles condition. Also see Chapter 17 for more information on macros.)

File ^FA

Another

Description

Opens a new window and loads the "next file to be loaded" into it. The name of the "next file to be loaded" is set by the File Read and File Open commands if they were given a file pattern with wildcards. (See Chapter 4 for further information on wildcards.)

Prompt

None.

Message

```
No more files to edit
```

Displayed if the "next file to be loaded" has not been set or if no more files match the pattern last given to a File Read or File Open command.

See Also

The File Open and File Read commands

Description

Saves the current file if it has been changed and closes the current window. If it has not been changed, then the window will just be closed. If other unhidden windows are open, the first unhidden window prior to the current window in the window list is made the current window. If there are no other unhidden windows, a new UNNAMED window is opened and made the current window.

Prompt

```
Save to which file?
```

If the current window is UNNAMED, you must enter a file name for the contents of this window.

Caution

```
That file exists.  Overwrite?
```

If the file named at the prompt above exists, CoEdit verifies that you wish to delete it before saving.

Message

None.

Restriction

Find All, Block Narrow

Description

Performs a File Done command on the current window but does not close the window. Instead, it prompts for a new file to load into the window. This command "exchanges" the current file for one on the disk. If no name is specified for the new file, this command is identical to File Done.

Prompt

```
Save to which file?
```

If the current window is UNNAMED, you must enter a file name for the contents of this window.

```
Read from which file?
```

After the current window has been saved, enter the name of the new file to be read in.

Caution

```
That file exists.  Overwrite?
```

If the file named at the first prompt above exists, CoEdit verifies that you wish to delete it before saving.

Message

```
File does not exist
```

Displayed if the file named at the first prompt above cannot be found on the disk. You may use CoEdit's ^L (locate) and ^F (file listing) to help locate the file.

Restriction

Find All, Block Narrow

Description

Discards the contents of the current window and loads a new file into it. If the current window has been changed, CoEdit prompts to verify that you wish to discard the modified contents.

Prompt

```
Read from which file?
```

Enter the file name to be loaded into the current window.

Message

```
File does not exist
```

Displayed if the file you named at the prompt above cannot be found on the disk. You may use CoEdit's ^L (locate) and ^F (file listing) to help locate the file.

Caution

```
File has been altered.  Ok to quit?
```

Displayed if the current window has been changed. Enter N to cancel the command or Y to continue.

Restriction

Find All, Block Narrow

Description

Merges the text of a disk file into the current window by inserting it at the current cursor position.

Prompt

```
Read from which file?
```

Enter file name whose contents are to be merged.

Message

```
File does not exist
```

Displayed if the file named at the prompt above cannot be found on the disk.

See Also

"Inserting blocks of text" in Chapter 7 for information on how the text of the disk file will be inserted into the current window.

Restriction

Find All, Read Only, Block Narrow

File ^FN

Next

Description

Saves the contents of the current window if it has been modified and loads the "next file to be loaded" into a new window. The current window is closed. The name of the "next file to be loaded" is set by the File Read and File Open commands if they were given a file pattern to load with wildcards. (See Chapter 4 for further information on wildcards.)

Prompt

```
Save to which file:
```

If the current window is UNNAMED, you must enter a name for the window contents before they can be saved on the disk.

Message

```
No more files to edit
```

Displayed if the "next file to be loaded" has not been set or if no more files match the pattern last given to a File Read or File Open command.

See Also

File Open and File Read

Restriction

Find All, Read Only, Block Narrow

File ^FO

Open

Description

Opens a new window and loads a file into it. If wildcard characters are used, the command loads the first file that matches the pattern and sets the name of the "next file to be loaded" for use by the File Another and File Next commands.

Using File Open to open a non-wildcard file name after using File Open to read wildcard file names will not interrupt the multiple wildcard file reading. If you do use the wildcard feature of this command, then the current multiple file reading (if any) is interrupted, and the new one is used. (See Chapter 4 for further information on wildcards.)

Prompt

```
Read from which file?
```

Enter a file pattern identifying the file(s) to be loaded.

Message

```
File does not exist
```

Displayed if file(s) named at prompt above cannot be found on the disk.

See Also

File Another and File Next

Description

Opens one or more new windows and loads one or more files into them. The number of windows opened depends on the file pattern input. If wildcard characters are used, the command loads as many files that match the pattern as it can and sets the name of the "next file to be loaded" for use by the File Another and File Next commands. (See Chapter 4 for further information on wildcards.)

If, at the end of the command, there are more windows open than are allowed on the screen, some windows will be hidden. They will reappear when the number of open windows decreases.

Prompt

```
Read from which file?
```

Enter a file pattern identifying the file(s) to be loaded.

Message

```
File does not exist
```

Displayed if file(s) named at prompt above cannot be found on the disk.

See Also

File Another and File Next

File ^FS

Save

Description

Saves the contents of the current window without closing the window. The current file name is the default save name, but any name may be entered. After the command executes, editing continues.

Prompt

```
Save to which file?
```

Enter file name into which window contents should be saved.

Caution

```
That file exists. Overwrite?
```

If the file named at the prompt above exists on the disk, CoEdit verifies that you wish to delete it before saving.

Restriction

Find All, Read Only, Block Narrow

Description

Saves the contents of all changed windows, optionally closes all windows, and opens an UNNAMED window. This allows quick updating of all modified files and a return to one open window. Files in hidden windows are included in update processing.

Prompt

```
Quit windows after saving to disk?
```

After the changed files have been written to the disk, you may instruct CoEdit to close the windows. Enter a "Y" here to do so, or an "N" to retain all current windows.

```
Saving contents of window nnnn:

Save to which file?
```

Each UNNAMED window must be written to the disk with a file name. If you do not enter a name at this prompt, the command will be canceled.

Caution

```
That file exists. Overwrite?
```

If a file named at the prompt above exists on the disk, CoEdit verifies that you wish to delete it before saving.

Message

```
Subwindows are open
```

This command cannot be used if there are find-all or narrow windows open. Use the Window List command to determine which windows have subwindows and the Block Widen and the Locate Undo commands as appropriate to complete editing these windows.

Restriction

Find All, Block Narrow

Chapter 12: Goto Commands

Using Goto Commands 12-1
Goto Begin Block 12-2
Goto Column 12-3
Goto End Block 12-4
Goto Insert Mode 12-5
Goto Line 12-6
Goto Match 12-7
Goto Overtype 12-8
Goto Previous 12-9
Goto Text Marker 12-10
Goto Window 12-11

Chapter 12: Goto Commands

Using Goto Commands

The Goto commands allow you to move quickly to another part of the current file or jump to another window. You can move to a specific line or window or to one of several markers. (Those markers can be block markers (see Chapter 7) or text markers (see Chapter 21). You can also use the Goto commands to switch the insert and overtype editing modes.

Goto ^GB

Begin Block

Description

Moves the cursor to the begin block marker, if it has been set. If the marker has not been set, a message is displayed on the window border.

If the marker is visible in the window, then the screen will not be redrawn; the cursor will simply move to it. If the marker is not visible in the window, the screen will be redrawn, displaying the current cursor line in the middle of the window.

Prompt

None.

Message

```
No begin block marker set
```

Description

Moves the cursor to the specified column on the current line. If the column is beyond the current end of the line, the line will be padded with spaces to the desired column. The cursor will then rest on the requested column.

Prompt

```
Goto which column?
```

Enter the column number you wish to go to.

Message

```
Invalid column number
```

The column number you entered is invalid. The number must be from 1 to 255.

Restriction

Read Only

Goto ^GE

End Block

Description

Moves the cursor to the block end marker, if it has been set. (If it has not been set, a message is displayed on the window border.)

If the marker is visible in the window, then the screen will not be redrawn; the cursor will simply move to it. If the marker is not visible in the window, the screen will be redrawn, with the marker in the middle of the screen.

Prompt

None.

Message

```
No end block marker set
```

Description

Forces CoEdit into insert mode, regardless of the current text entry mode. This command is very useful for macros, since the text entry mode cannot be queried during macro execution.

Another way of forcing CoEdit into the insert mode is use of the co.cml macro language condition Insert. See Chapter 6 for a description of the CoEdit Macro Language.

Prompt

None.

Message

None.

Goto ^GL

Line

Description

Moves the cursor to a particular line in the current window. Line numbers are sequentially numbered from 1 at the beginning of the file.

Prompt

```
Enter line number:
```

Enter the line number to go to, in range from 1 to the total lines in the file, which is displayed on the status line.

Message

```
Bad or invalid line number
```

You must enter a valid integer number in the range 1 to the total number of lines in the window.

Description

Moves the cursor to the character "matching" the character under the cursor. Only the characters (,), {, }, [, and] are considered by this command.

If the cursor is on an opening character (that is, (, {, or [) then CoEdit searches forward for the corresponding closing character forward in the file. If the cursor is on a closing character, CoEdit searches for the corresponding opening character backwards in the file. If your file is a program source file, you should be aware that comments are not excluded from the search.

If the cursor is not on one of these six characters or if the matching character can not be found, CoEdit beeps (if environment option BEEP TONE is On).

Prompt

None.

Message

None.

See Also

Environment Configure for setting the environment parameters SHOWMATCH and BEEP.

Goto ^GO

Overtype

Description

Forces CoEdit into overtype mode, regardless of the current text entry mode.

Another way of forcing CoEdit into the insert mode is use of the co.cml macro language condition Insert. See Chapter 6 for a description of the CoEdit Macro Language.

Prompt

None.

Message

None.

Description

Moves the cursor to its previous position in the current window, prior to the execution of the last command.

Prompt

None.

Message

None.

Goto ^GT

Text Marker

Description

Moves the cursor immediately to the right of a specified text marker, 0 through 9, if the marker has been set. If the input marker has not been set, a message is displayed on the window border.

Prompt

```
Goto which marker?
```

Enter a single digit 0-9 of the marker to move to. Non-digits are ignored.

Message

```
Marker not set
```

Displayed if the selected marker has not been set with the Text Set Marker command.

Description

Displays a list of open windows and prompts for the number or short name of the new current window. If no window is selected, the current window remains unchanged.

Prompt

```
Window to go to:
```

Enter the number or short name of a window listed below the prompt. See the Window List command in Chapter 23 for a description of the window input screen.

Message

```
Invalid window choice
```

Displayed if a non-existent window number or short name is entered.

See Also

Window Assign for information on giving short names to windows.

Chapter 13: Internal Commands

Using Internal Commands 13-1
Internal Character Value 13-2
Internal Delete Buffers 13-3
Internal Key Code 13-4
Internal Table of Characters 13-5

Chapter 13: Internal Commands

Using Internal Commands

The Internal commands in CoEdit perform low-level maintenance and utility functions.

For example, the Internal Delete (^ID) command allows you to delete your delete buffers from memory. This command is especially useful if you use functions such as Block Program, Block Filter, and Block Narrow/Widen, because large pieces of unwanted deleted text may be needlessly taking up memory.

In addition, the Internal command group contains several very useful programming commands, including key scan code information, character value conversions, and the ASCII character table.

Internal ^IC

Character Value

Description

Displays the ASCII value of the character below the cursor. The value will be displayed on the window border.

Prompt

None.

Message

None.

Description

Removes deleted text from the garbage stack, freeing up memory for other program functions. You can select individual buffers to free (that is, delete) or A for all of them.

If you have requested CoEdit to save a large number of deleted buffers (see the environment option Undo Buffers Max in Environment Configure, Chapter 10), there is likely to be something in there that you do not want. CoEdit can perform its operations more quickly when there are fewer delete buffers.

You should use this command carefully: once the buffers are deleted from the garbage stack, they cannot be recovered.

Prompt

```
Buffer to delete or A for all?
```

Enter the number of a valid delete buffer or A to delete all of them.

Message

None.

Internal ^IK

Key Code

Description

Displays the scan code of a key on the keyboard. Note that for ten keys the scan code that CoEdit displays is not the actual scan code for the computer. There are a few keys whose scan codes CoEdit modifies in order to recognize them. Specifically, they are the keys Shift0 through Shift9, which produce only a 0-9 on standard input.

Note that on UNIX systems, this scan code is emulated, as this information is not normally available to applications.

Prompt

```
Strike a key:
```

Strike a key on the keyboard. The name of the key will be displayed, as well as the scan code for the key to the right of the prompt. Press ABORT to exit the command.

Message

None.

See Also

Environment Macro for information on using these scan codes for redefining the macro begin and macro end keys.

Description

Displays an ASCII table of characters and their hexadecimal equivalents.

Prompt

```
Press the PageUp, PageDown, or ABORT keys
```

Press PageUp or PageDown to see the other screen, or press ABORT or Ctrl-C to terminate the command.

Message

None.

Chapter 14: Jump Commands

Using Jump Commands 14-1
Jump Back Half Page 14-2
Jump Down Line 14-3
Jump End of File 14-4
Jump Forward Half Page 14-5
Jump Top of Screen 14-6
Jump End of Screen 14-7
Jump Next Page 14-8
Jump Previous Page 14-9
Jump Scroll Lock 14-10
Jump Top of File 14-11
Jump Up Line 14-12

Chapter 14: Jump Commands

Using Jump Commands

The Jump commands "jump" the cursor to another part of the current file. In most jump commands, the cursor will move to a new line in the file and the contents of another part of the window will appear on the screen. Two jump commands keep the cursor on the same line, if possible: Jump Up and Jump Down shift the window contents up or down one line. If the cursor is at the top or bottom of the window, it will remain there; otherwise it will move up or down the screen with the current line.

It is possible to jump quickly to the top or end of the current file or forward or backward a half or full page. The "page size" of the current window is defined as its height in lines. With one window open, the page size is usually 22. "Half page" commands will then move the cursor 11 lines forward or backward in the file.

Here are the Jump commands arranged by the direction of the cursor movement:

Moving backward

```
^JU Shift window lines up
^JB Go back half page
^JP Go to previous page
^JH Shift to home line
^JT Go to top of file
```

Moving forward

```
^JD Shift window lines down
^JF Go forward half page
^JN Go to next page
^JL Shift to last line
^JE Go to end of file
```

Jump

^JB or Ctrl-PgUp

Back Half Page

Description

Moves the cursor backwards one half page in the file and redisplays the window, keeping the cursor at the same line in the window if possible. If there are fewer than a half-page number of lines to the top of the file, the cursor moves to the first line of the file and the first line of the window.

Prompt

None.

Message

None.

Description

Shifts the contents of the window down one line and moves the cursor up one line. This command scrolls new lines into the window without changing the cursor's line number. If the cursor is on the top line of the window, it is moved to the next line in the file as that line scrolls up. The command has no effect if the cursor is on the last line of the file.

Prompt

None.

Message

None.

Jump

^JE or Ctrl-End

End of File

Description

Moves the cursor to the last character in the file and redisplays the window if necessary.

Prompt

None.

Message

None.

Description

Moves the cursor forward one half page in the file and redisplays the window, keeping the cursor at the same line in the window if possible. If there are fewer than a half-page number of lines to the end of the file, the cursor moves to the last line of the file and the last line of the window (unless the file is smaller than the window).

Prompt

None.

Message

None.

Jump

^JH or Shift-Home

Top of Screen

Description

Attempts to take the current text line and place it on the first line of the window. This command, however, will not allow you to bring the last line of the file above the last line of the window. If there are fewer lines in the file than there are lines in the window, this command has no effect.

Prompt

None.

Message

None.

Description

Takes the current line and places it on the last line of the window. If the number of lines above the cursor is less than the number of lines in the window, then the screen is only shifted down that many lines, or zero if there are none.

Prompt

None.

Message

None.

Jump ^JN or PgDn

Next Page

Description

Moves the cursor forward for one full page in the file and redisplays the window, keeping the cursor at the same line in the window if possible. If there are fewer than a full-page number of lines to the end of the file, the cursor moves to the last line of the file and the last line of the window (unless the file is smaller than the window).

Prompt

None.

Message

None.

Description

Moves the cursor backwards one full page in the file and redisplays the window, keeping the cursor at the same line in the window if possible. If there are fewer than a full-page number of lines to the top of the file, the cursor moves to the first line of the file and the first line of the window.

Prompt

None.

Message

None.

Jump ^JS

Scroll Lock

Description

Toggles the scroll lock flag, which is used for simultaneous window action in the Jump commands. This is provided for keyboards that do not have a Scroll Lock key on the keyboard. If this flag is Off, then the keyboard's scroll lock status is used; if there is no Scroll Lock key, then the status is off. If this flag is On, then the Scroll lock flag is on, regardless of the status of the keyboard.

Prompt

None.

Message

None.

See Also

CML conditional In_Scroll_Lock

^JT or Ctrl-Home

Jump
Top of File

Description

Moves the cursor to the first character of the file and redisplays the window.

Prompt

None.

Message

None.

Jump

^JU or Shift-Up Arrow

Up Line

Description

Shifts the contents of the window up one line and moves the cursor up one line. This command scrolls new lines into the window without changing the cursor's position. If the cursor is on the bottom line of the window it is moved to the previous line in the file as that line scrolls down. The command has no effect if the cursor is on the first line of the file.

Prompt

None.

Message

None.

Chapter 15: Key Commands

Using Key Commands 15-1
Key Commands 15-2
Key List 15-3

Chapter 15: Key Commands

Using Key Commands

The Key commands are provided to display CoEdit's commands and function key bindings. To change the definition of the key bindings, you simply need to change the co.nfy file. See Chapter 10, Environment Commands, for a complete description of the co.nfy file and how to change it.

Key ^KC

Commands

Description

Displays all of CoEdit's commands regardless of current key bindings. This is useful for determining key bindings and getting a quick overview of CoEdit commands.

Prompt

```
Press the PgUp, PgDn, or ABORT keys
```

Message

None.

Description

Lists the definition of the 40 function keys (Normal, Shift, Ctrl or Alt F1 - F10). Shows the description of their key bindings or the macro descriptions as appropriate. (Not all of these keys may be available on your keyboard. See your terminal manual for further information.)

Prompt

```
Press ABORT to continue
```

Press ABORT when you are done viewing the screen.

Message

None.

Chapter 16: Locate Commands

Using Locate Commands

The Locate commands include powerful locate and replace functions. All of the commands accept normal strings and "patterns" (such as regular expressions, as described in the following section). The two basic commands in this group are Locate String and Locate Replace; each of these commands has an extensive option input menu.

Locate Next and Prev are quick versions of Locate String that do not require option input. Locate Again repeats the last locate or locate and replace command.

Entering Locate Options

The operation of the Locate String and Replace commands has two steps. First, you are asked for the string or pattern to find or replace. Then you are presented with a set of options. To choose an option, press the first letter of its name. Thus to find the first occurrence of a string, press F at the options prompt. The default option, Next, will be unhighlighted and First will be highlighted. The same principle holds for all other options.

The locate options menu looks like this:

```
Which:      Next      First  Last  All  -All  #:
Direction:  Up        Down
Case:       Consider  Ignore
Lines:      Range:
```

For replace there is another option:

```
How:        View      Prompt     Quick
```

Which

This option controls where the search will begin and how many correct matches should be found before ending the command. The Next option is the default; it means start from the current cursor location. First means start from the first character in the file, Last from the last character in the file.

The # option for Locate commands allows you to input the number of occurrences from the cursor location should be found: it finds the *n*th match. For Locate and Replace commands, it tells how many replacements should be made.

The All option for the Locate and Replace command means begin at the top of the file and change every occurrence. See the section "The All Choice: Find All Mode" later in this chapter.

Direction

CoEdit supports searching in either direction from the cursor location. Up means towards the top of the file, Down towards the bottom. This option is only meaningful with the Next and # options. First and All default to Down; Last defaults to Up.

Case

You can determine whether CoEdit will Consider case or Ignore case during searches. If case is considered, then the input string/pattern must match the file text exactly. If case is ignored, then the characters A-Z and a-z will match each other.

Lines

You can restrict the scope of the search or search and replace by specifying a line range in this option. All or A when replacing means the entire file. (For searching only, it means something else. See the section "The All Choice: Find All Mode" for further information.) Otherwise the range must be of the form #-#, where each number is between 1 and the number of lines in the file. T and B can be used for the Top and Bottom lines of the file, as in 132-B or T-100.

How

This option is only available during search and replace functions. It allows you to specify what kind of screen display and prompting should take place during the replace operation. The Quick option tells CoEdit to perform all changes without updating the screen or prompting. The View option will cause CoEdit to highlight each match of the search text on the screen. If you strike a key while CoEdit is displaying the changes, it will complete the changes without displaying them. The Prompt option will display each match before replacing it and prompt you to verify that it should be replaced. You can enter Yes or No, enter Quick mode, or interrupt the command.

The All Option: Find All Mode

A very useful feature in the Locate String command is called Find All. With use of this global locate function, the All option of the Locate String command will gather up all lines that contain the input string/pattern and create a subwindow. (See Chapter 4 for a complete description of subwindows.)

Similarily, the -All option (read: "find not all") will gather all those lines that do not contain the search string/pattern. (References in this manual to "find all" apply equally to the find all or the find not all options, unless otherwise indicated.)

The contents of the lines in the resulting subwindow generated by either find all option may be changed, but they cannot be deleted or moved in any way, since they are mapped to specific lines in the parent window. The Find All level is displayed on the window border in square brackets ([ ]) after the file name.

To return to the parent window, the Locate Undo command is used. All changes in the subwindow will be reflected in the parent window.

It is possible to nest subwindows with successive Find All commands and to "pop" out of Find All levels with the Locate Undo command. This ability to nest Find All windows is very useful for creating a window containing exactly what you want.

When you Undo a find all level, you are returned to the parent window and placed on the line that you were on in the previous level. Thus you could find all occurrences of a variable, spot an incorrect assignment

statement, and pop back to the parent window with the context of that line on the screen.

Using Regular Expressions

CoEdit allows you to use operating system regular expressions to locate text. See your operating system documentation for a description of regular expressions (often found under ed or sh). When replacing text, you may search using a regular expression, but the replacement is treated as only a simple string (that is, none of the special regular expression characters has any meaning).

Chapter 17: Macro Commands

Using Macro Commands 17-1
Defining a Keystroke Macro 17-1
- Temporary Macros 17-2
- Permanent Macros 17-2
- Ending Keystroke Macro Recording 17-2

CML Macros 17-3
- Defining a CML Macro 17-3
- CML Macro Commands 17-3
- Editing a Macro 17-4
- Executing and Terminating a Macro 17-5
- Searching for Macros on the Disk 17-5
- Changing and Adding Macro Key Bindings 17-5
- Debugging Macros 17-5

Macro Compile 17-6
Macro Define 17-9
Macro End Definition 17-12
Macro Flush 17-13
Macro Go 17-14
Macro Key Name 17-15
Macro List 17-16
Macro Save 17-18
Macro Trace 17-20
Macro Uncompile 17-23

Chapter 17: Macro Commands

Using Macro Commands

The Macro commands are used to create, modify, and list CoEdit macros. Macros are sequences of keystrokes and CoEdit commands that can be replayed over and over in order to greatly reduce the time and effort required to perform repetitive or long editing chores.

A macro may be bound to almost any key sequence or key (except those that generate an alphanumeric character like A or +). CtrlA, Shift0, ^KX and AltM are all examples of keys to which macros may be bound. (See Appendix A for a complete list of individual keys available for binding to macros.)

To execute a macro, press the key or key sequence it is bound to. To see a list of macros that are currently bound to keys, use the Macro List or the Key List command. Macros may also be executed via the Miscellaneous Execute command. (See Chapter 26, Miscellaneous Commands, for a complete description of this command.)

There are two kinds of macros in CoEdit: keystroke macros and CML (CoEdit Macro Language) macros. (For a complete discussion of macros, see Chapters 5 and 6.)

Keystroke macros are a subset of CML macros and are simply sequences of keystrokes that can be replayed later. They can be created either by recording directly from the keyboard or by creating the source code for the macro in ASCII form and compiling it into raw keystroke form.

Defining a Keystroke Macro

The first way to create keystroke macros is with the Macro Define command F9. CoEdit will ask you whether the new macro should be a temporary or a permanent macro.

Temporary Macros

Temporary macros are macros which are, generally speaking, intended to repeat a series of keystrokes for quick, one-time editing tasks and then be tossed away. Experience has shown that many macros are used simply to perform a specific task one time, and therefore do not need to be saved from one session to another.

A temporary macro is no different from any other macro, except that its name is always TEMP (and is therefore placed in a macro file called TEMP.cmf) and its description is "Temporary". Every time you create a temporary macro, the existing temporary macro (TEMP.cmf) will be overwritten. Temporary macros will be created in the same directory as other macros, as dictated by the Environment Macro command (see Chapter 10).

As a convenience, CoEdit provides a way to save temporary macros if you should decide that the macro will be of use in the future. The Macro Save command (see details below) will allow you to specify a new name for the temporary macro and enter a new description for the macro.

Permanent Macros

Permanent macros are macros that you may want to keep for a while to perform the same task in the future. If at the Macro Define command you indicate that this keystroke macro is not temporary, CoEdit will prompt for the name of the macro you wish to create, the key to which the macro will be attached (if it is an individual key like F10), and for a short description of what the macro does.

After answering the questions at the Macro Define command, you will return to normal CoEdit operation, except that now all of your keystrokes are being saved to a macro disk file.

Ending Keystroke Macro Recording

Keystroke macro definition is ended with the Macro End Definition command. The new macro file will be closed and saved. The macro is now defined. To execute the macro, strike the key to which it is bound, or use the Miscellaneous Execute command to execute the macro by name. (See Chapter 25 for Miscellaneous Execute, and the following section for detailed instructions on macros.)

You may occasionally make mistakes while recording a macro. For this reason, CoEdit allows you to edit macros. (See the section "Editing a Macro" later in this chapter.)

CML Macros

CML is the CoEdit Macro Language. It allows you to create more "intelligent" macros using Do, For, While, and If programming structures to test conditions and create program loops. (See Chapter 5 for a complete description of the CML language.)

Defining a CML Macro

You can create source code for CML macros by simply typing in the macro in source form, just as you would any other programming language.

Macro source code is free-format; the only restriction is that the first two lines must start with an asterisk. They define the name of the macro and supply a comment or description of the macro's function. For example,

```
*SAVETEMP
*Save window to temporary file
File_save CtrlB "TEMP" Enter
```

In this example, SAVETEMP is the name of the macro, and the second line describes its function. See the description of Macro Compile (^MC) later in this chapter for further information on compiling macros and macro source file formats.

CML Macro Commands

Most CML commands must be typed in explicitly while editing the macro, because there is no way to "record" them in macro define mode. There are four special CML commands, however, that can alter the execution of a macro. They may actually be encoded during keystroke recording or by explicitly editing them into the macro source.

During the recording of a macro, you may press F9 again. This will cause CoEdit to ask you for an embedded macro option. There are four of these options:

G	Go	Turn off tracing
T	Trace	Turn on tracing
V	Variable field	Request keyboard input
W	Wait	Pause macro execution

The Trace option will turn on macro tracing at that point when the macro is replayed. (See Macro Trace for more information.)

The Go option will turn tracing off. (See Macro Trace for more information.)

The Variable field option will suspend macro execution until a carriage return is read from the keyboard. This allows you to ask for input from the keyboard.

The Wait option will suspend macro execution for approximately five seconds.

To encode these options into source macro format, use the following syntax:

```
Varfield            ; Request input
Wait                ; Pause macro
```

Or alternatively

```
Alt= "V"            ; Request input
Alt= "W"            ; Pause macro
```

Editing a Macro

CoEdit allows you to edit your macros by converting the coded format of the disk macro (the object code) into an understandable English-style macro format. This process is called uncompiling, and is done with the Macro Uncompile command (^MC).

After a macro has been uncompiled, you are free to edit the macro. When you are satisfied with your changes, you may encode it once again by compiling it into executable CML object code, using the Macro Compile command (^MC). For more information, see the Macro Compile and Macro Uncompile commands.

Executing and Terminating a Macro

Macros can "call" other macros; that is, they can refer to keys that call other macros. The actual number of levels will vary depending on the number of macros stored in memory and the number of files CoEdit currently has open. Macros can call themselves recursively, which means one of the commands in the macro says "execute this macro again". To cancel a macro, press Ctrl-Break or press the Ctrl and Alt keys simultaneously until the macro stops.

Searching for Macros on the Disk

CoEdit searches for macro files in two places: the current directory and the macro library directory. The library directory can be set in the environment and stored in a configuration file. Each time CoEdit enters a new working directory (see System Goto command), it identifies the available macro files after reading the configuration file. This means macros have priority over key definitions in the configuration file co.nfy. (For example, if F1 is defined as Window Open in the configuration file and a macro exists for F1, pressing F1 will execute the macro.)

Changing and Adding Macro Key Bindings

Once a macro is defined, you may bind it to one or more keys or key sequences by editing the co.nfy file. This file contains all CoEdit command bindings. (See the Chapter 10 concerning Environment Commands.)

Debugging Macros

CoEdit has a built-in source-level macro debugger. With it, you can trace the execution of a macro, delete commands, insert keystrokes, and trap breakpoints. See the Macro Trace command later in this chapter for a complete description.

Macro ^MC

Compile

Description

Compiles a macro source file (.cms for CoEdit Source File) into a coded macro file (.cmf for CoEdit Macro File). The source must be in the current window. If the window is unnamed, you will be asked to supply a name. The first line of the window must start with an asterisk and be immediately followed by the name of the macro to be defined. The second line must also start with an asterisk. It should be followed by a description of the macro for identification purposes, though the description is not required. The third line may contain, if you so desire, the compiler directive #debug, which will cause debugging information to be embedded into the macro. The source code for the macro must begin on the next line and may be as long as is needed.

Here is an example:

```
-------------------------------------
|*DIRLIST
|*Dir listing asking for pattern
|"Hello" List_file Varfield PgDn PgDn
|
```

The above macro will type "Hello" and then invoke the System List command. Macro execution will be suspended until the operator enters a file pattern and strikes ENTER. Then the macro will instruct CoEdit to move two pages down in the display.

The case (upper or lower) of the symbols does not matter. Thus "Enter" can be typed as "enter" or "ENTER".

The symbols must be separated by white space (a space, a tab, or a carriage return). Text must be enclosed within double quotes, although single letters may be entered with just white space around them. Symbols need not be separated from a quote. Quotes may be entered by escaping them with the escape character backslash (\). A backslash may be entered by typing two in succession. Any other character when escaped yields simply the character itself (that is, \a is the same as just a). Any lines that begin with a semicolon are comment lines (that is, the information on them is ignored).

When the macro is successfully compiled, the object macro code is saved on the disk in a file by the same name, with the extension .cmf.

Prompt

None.

Message

`Compilation had` *X* `warnings and` *Y* `errors.`

coC, the CoEdit macro compiler, found errors in the source file, due to unrecognized keywords, bad constructs, etc. The cursor is placed on the first line that got an error or warning. Correct the error(s) and recompile.

`Compilation successful`

CoEdit has compiled the macro successfully and the macro has been stored on the disk. The macro is ready to execute.

Restriction

Find All, Read Only, Block Narrow

See Also

Macro Uncompile for further information on translating coded macro files into source form.

See also Chapters 5 and 6 for further information on macro commands and functions.

Description

Initializes a disk file to accept keystrokes. The disk file will be created in the current directory, unless the environment variable Default to Library is On. (See the Environment Macro command in Chapter 10.) First CoEdit will ask you if this is a temporary macro. (For a complete description of temporary macros, see the section "Temporary Macros" earlier in this chapter.) If it is not temporary, you will be prompted for the name of the macro, an optional (though recommended) short description of what the macro does, and the key to which the macro will be assigned. During macro definition the word MACRO flashes on the status line.

While defining a macro, the F9 key, which can be configured with the Environment Macro command, will invoke a special menu of macro subcommands. The commands available are G (Go), T (Trace), V (Variable field), and W (Wait).

Go will cause a Macro Go command to be executed when it is encountered during the execution of the macro.

Trace will invoke the Macro Trace command.

Variable Field is a powerful addition to macros. It allows them to be suspended while waiting for information (some response or command) to be entered. CoEdit will continue when the ENTER key is struck.

Wait will suspend the execution of the macro for approximately five seconds. This will allow you to see something on the screen before the macro continues execution.

Prompt

```
Is this a temporary macro?
```

You must tell CoEdit whether or not this is a temporary macro.

```
Enter macro name:
```

Enter the name of the macro you will define. It can be from one to eight characters in length, and may contain any valid operating system file name characters.

```
Enter macro description:
```

Enter a short description (30 characters or less) of what this macro does. This optional description will be stored in the macro file; it will appear if the file is typed by operating system command or by the System Type command and will be displayed in menus and by the Macro List and Key List commands.

```
Strike the key to which this macro should
be bound:
```

If you want this macro to be bound to a single keystroke (such as a function key), enter that keystroke now. If you press ABORT, then you will proceed to define the macro, but it will not be bound to a key sequence. Note: this assignment is not permanent unless you update the environment to disk.

```
Select:  Go  Trace  Var Field  Wait
```

If you press the macro define key (default: F9) during macro definition mode, you may enter G for Go, T for Trace, V for Variable field and W for Wait.

Caution

```
This key is defined.  Redefine it?
```

If the key pressed at the prompt above already has a definition, CoEdit verifies that you wish to delete the key's current meaning.

See Also

Environment Update/Save in Chapter 10 for a description of how to save your macro key bindings.

Macro ^ME or F10

End Definition

Description

Ends the current macro definition and closes the disk file. Note that unlike ^ME, Alt- may be used anywhere, including at prompts.

Prompt

None.

Message

```
Not in macro define mode
```

This command can only be used to end a macro definition begun with the Macro Define (or F9) command.

Restriction

Not Defining a Macro

Description

Flushes all macros from CoEdit's memory. The macro disk files are unaffected. Keys that were previously macro-defined will continue to be macro-defined; however, the next time the macro is invoked, it must be read from the disk. This is useful if you wish to use a new version of a macro you have already used, because it forces CoEdit to reread the macro file.

Prompt

None.

Message

None.

Macro ^MG

Go

Description

Turns off macro tracing. This command may be embedded within macro definitions to turn off tracing. When this command is used in conjunction with the Macro Trace command, selected sections of macros can be traced and debugged.

Prompt

None.

Message

```
Macro tracing OFF
```

See Also

Macro Trace for further information on the Macro Trace command.

Description

Inserts the name of a command bound to a key sequence into the current window at the current cursor position. This is useful when you are writing macros and you do not know the name of a command. For example, assume the Goto Line function were bound to the key sequence ^GL. If you execute the Macro Key Name function and enter the key sequence ^GL, then CoEdit will insert Goto_line into the text. If there is a command bound to the F1 key, for example, you may strike that as well.

If you enter a key sequence that is not bound to any command, CoEdit will beep.

Prompt

```
Enter key sequence:
```

Enter the key sequence that is bound to the command whose name you want.

Message

None.

Restriction

Read Only

Macro ^ML

List

Description

Lists all currently defined macros on the screen. An example of the output of this command is given below. These are macros that CoEdit knows about either because it has executed them, or because they are bound to keys.

MACRO	BOUND TO	DESCRIPTION
BRACES	CtrlB	Finds braces
	F1	
STARTUP	ShiftF10	CoEdit start macro
TEMP	AltN	Temporary
ZONED	Backspace	[Not available]
SMKS	[None]	Macro for main()
BAD	[N/A]	This macro is OK

In the event that a macro is bound to more than one key, such as BRACES in this example, all key bindings beyond the first binding (for example, F1) are listed immediately below the first binding. The description of the macro is displayed to the right of the key bindings.

Macros are displayed with the High Intensity attribute (see Environment Attribute) if they reside in the current directory, or with the Low Intensity attribute if they are from the macro library. In our example, BRACES, STARTUP, ZONED, and SMKS are in the macro library. TEMP and BAD are in the current directory.

An asterisk immediately preceding a key binding indicates that that macro is currently in memory. In this example, F1 has been used to invoke macro BRACES, and the Miscellaneous Execute (^ZX - see Chapter 26) was used to execute the macro SMKS.

If for some reason the macro file for a key cannot be opened (for example, if it was deleted), the message Not available is displayed next to its bindings, as is the case for ZONED in our example.

If a macro is not bound to any key, its key bindings will display as None. This can only happen if the Miscellaneous Execute command was used to invoke the macro. In our example, SMKS is not bound to any key.

Finally, if the macro compiler table file co.tab cannot be opened, CoEdit will display N/A, meaning that the file was not available. In the example above, the bindings for macro BAD could not be determined.

Prompt

```
Press the PageUp, PageDown, or ABORT keys
```

Press the PageDown key to see the next page of macro listings, PageUp to see the previous page, and ABORT to return to normal editing.

Message

None.

See Also

The following sections in Chapter 10: Environment Attribute for information on setting the High and Low Intensity screen attributes, the Environment Macro for information on setting the Macro Library, and Environment Read for information on affecting new key bindings.

Macro ^MS

Save

Description

Saves the current temporary macro into another file and allows you to change its description and key binding. This command is useful if you decide that you really want to keep the temporary macro you defined. The macro file is renamed, its description is changed, and its key binding is modified.

If the temporary macro is in memory, then the key bound to that macro is still operational. When macros are flushed, either explicitly by the operator or internally by CoEdit, the key previously bound to the old temporary macro will generate the message "Macro file missing" (because it has been renamed).

Prompt

```
Enter macro name:
```

Enter the new macro name for the temporary macro. This name should be from one to eight characters in length, should be a valid operating system name, and should not contain a period.

```
Enter macro description:
```

Enter a new description for the macro.

```
Strike the key to which this macro should
be bound:
```

Enter the key that you want to have invoke the macro. If you press ABORT, then the macro will not be bound to any new key. Pressing ABORT will not cancel Macro Save.

Message

```
File exists
```

The name you entered is the name of an existing macro file. Try another name.

Macro ^MT

Trace

Description

Turns on the trace mode for macros. This mode allows you to debug your macros by single-stepping through them instruction by instruction.

In order to debug a macro, the macro must have been compiled using the #debug coC compiler directive or by specifying -d on the CMC command line. This causes debugging information to be included in the object macro file. Then you must issue the Macro Trace command to enable macro debugging in CoEdit. This way you can debug only the macros you want, while you let the other macros run freely.

When you debug a macro, a debugging window appears on the screen. The source to the macro appears on the bottom border of the window. For example, the following line might appear:

```
0003:If Validblock - F
```

This means that you are viewing source from line three of the macro source file; the source for that line is "If Validblock"; and that the value of the condition (Validblock) is False (that is, there is not a valid block defined).

In addition to this information, a pop-up debugging menu appears on the screen. It presents you with the following choices:

G	Go. This means that the macro may continue untraced. Control of CoEdit will return to the keyboard when macro execution terminates, or when the macro encounters a breakpoint.

Space	Trace single instruction. This lets the current instruction execute.
ABORT	Terminate. This terminates macro execution immediately.
Ins	Insert keystroke. By hitting the Insert key, you are able to insert keystrokes into the macro (but only for this execution).
Del	Delete instruction. Hitting the Delete key causes CoEdit to skip the current instruction and pass on to the next.
Up	View previous instruction. By hitting the UP arrow on the keyboard, you can view the previous instruction.
Dn	View next instruction. By hitting the Down arrow on the keyboard, you can view the next instruction.
I	Same as Ins.
D	Same as Del.
-	Same as Up.
+	Same as Dn.

Prompt

None.

Message

```
Macro tracing ON
```

See Also

Macro Go for further information on turning off macro tracing and Macro Define for further information on defining macros and embedding commands within them.

Description

Uncompiles a disk macro. The macro must currently be defined. The uncompiled macro source will be put into a new window for editing. This window is a normal window, which may be saved or edited in any fashion.

Prompt

```
Enter name of macro to uncompile
```

Enter the name of the macro you wish to uncompile.

Message

```
Macro file missing
```

No macro exists with the name you have entered.

See Also

Macro Compile for further information on translating the macro source into coded macro format.

Chapter 18: Print Commands

Using Print Commands 18-1
Print Again 18-2
Print Block 18-4
Print Cancel 18-5
Print Delete 18-6
Print Eject 18-7
Print File 18-8
Print Hold 18-10
Print Pause 18-12
Print Queue 18-14
Print Release 18-16
Print Terminate 18-17
Print Window 18-18

Chapter 18: Print Commands

Using Print Commands

The Print commands in CoEdit allow you to print to the system printer(s). By using the UNIX lp command, you can have CoEdit do all the nitty-gritty work for you. Just set up the initial options by using the Environment Printer command (see Chapter 10).

The exact functionality of these commands is affected greatly by the commands you configure with the Environment Printer command.

Print ^PA

Again

Description

Allows you to change the number of copies to be printed for one or more files on the print queue.

Prompt

```
Enter file name:
```

Enter the file pattern for those files you wish to change.

```
Enter number of copies:
```

Enter the new number of copies to be printed for the files named above.

```
Which printer?
```

Enter the printer name, for example: lpr.

Message

```
The print queue is empty
```

Displayed if there are no files in the print queue.

Printer not available

The selected printer is not available. Select another or check the environment.

Print ^PB

Block

Description

Saves the current block to a temporary disk file and adds that file to the print queue.

Prompt

```
Number of copies?
```

Input the number of times the block is to be printed.

```
Hold files?
```

Press Y to add the block to the queue so that it will not actually print until you release it.

Message

```
Block undefined or out of order
```

The begin and end block markers must be set, with begin before end in the file.

Description

Cancels the file currently being printed and removes it from the queue. The first unheld files on the queue will begin printing on the appropriate printers.

Prompt

```
Which printer?
```

Enter the printer name, for example: lpr.

Message

```
The print queue is empty
```

Displayed if there are no files in the queue.

```
Printer not available
```

The selected printer is not available. Select another or check the environment.

Print ^PD

Delete

Description

Allows you to remove one or more files from the print queue. Once removed, the files will not be printed. However, Print Delete will not remove the file that is currently being printed. To cancel the file that is currently printing, use the Print Cancel command.

Prompt

```
Which printer?
```

Enter the printer name, for example: lpr.

```
Name of file to delete from queue:
```

Enter the file pattern to remove from the queue. Press ^Q for queue list.

Message

```
The print queue is empty
```

Displayed if there are no files in the print queue.

```
Printer not available
```

The selected printer is not available. Select another or check the environment.

Description

Ejects (form feeds) the selected printer one page. The eject is queued as a normal file and is not performed immediately.

Prompt

```
Which printer?
```

Enter the printer name, for example: lpr.

```
Hold files?
```

Press Y to add the eject to the queue so that it will not actually print until you release it.

Message

```
Printer not available
```

The selected printer is not available. Select another or check the environment.

Print ^PF

File

Description

Adds one or more disk files to the print queue, prompting for the number of copies to be printed. CoEdit also asks whether files should be added in "hold" mode.

Prompt

```
Name of file to print:
```

Enter the file name of those files to be printed. The name of the file may contain wildcard characters. Press ^F to see the a listing of the disk directory.

```
Number of copies:
```

Enter the number of times that each file should be printed.

```
Hold files?
```

Press Y to add the file(s) to the queue in hold mode. These files will not actually begin printing until they are released.

```
Which printer?
```

Enter the printer name, for example: lpr.

Message

`File does not exist`

Displayed if file pattern input at first prompt does not match any existing disk files.

`Printer not available`

The selected printer is not available. Select another or check the environment.

Print ^PH

Hold

Description

Puts one or more queued files in hold mode. These files will not be printed until they are released using the Print Release command. Use print holding if, for example, you wish to execute several programs that use the printer with the External Execute command. By holding queued files, you can avoid CoEdit's generating printed output in between your executions.

Prompt

```
Which printer?
```

Enter the printer name, for example: lpr.

```
Name of file to hold:
```

Enter a file pattern selecting those files in the queue to be held. You may press ^Q to see the print queue. If the input pattern matches no files, the command has no effect.

Message

```
The print queue is empty
```

Displayed if there are no files in the print queue to hold.

Printer not available

The selected printer is not available. Select another or check the environment.

Print ^PP

Pause

Description

Pauses any current printer output until the same command is issued again; acts as a toggle for whether or not output is going to the printer.

Prompt

```
Which printer?
```

Enter the printer name, for example: lpr.

Message

```
The print queue is empty
```

Displayed if there are no files in the print queue.

```
Printing paused
```

Displayed if the Print Pause command turned pause on, stopping printing.

```
Printing resumed
```

Displayed if the Print Pause command turned pause off, restarting printing.

Printer not available

The selected printer is not available. Select another or check the environment.

Print ^PQ

Queue

Description

Displays the print queue with the following information:

- queue entry number
- H if entry is held
- F if entry is a disk file
- B if entry was a block print
- W if entry was a window print
- the number of copies
- the file name

The total number of files is at the top of the screen. The PageUp and PageDown keys can be used to scroll through the queue; pressing ABORT returns to normal editing.

Prompt

```
Press the PageUp, PageDown, or ABORT keys
```

Use the paging keys to view the queue and ABORT to end viewing.

Message

```
The print queue is empty
```

Displayed if there are no files in the print queue.

Printer not available

The selected printer is not available. Select another or check the environment.

Print ^PR

Release

Description

Releases one or more queued files from hold mode. Files that are held on the queue will not be printed until released. Files may be held when they are added to the queue and also with the Print Hold command.

Prompt

```
Which printer?
```

Enter the printer name, for example: lpr.

```
Name of file to release for printing:
```

Enter a file pattern selecting those files in the queue that should be released. If the specified pattern does not match any queued files, the command has no effect.

Message

```
The print queue is empty
```

Displayed if there are no files in the print queue.

```
Printer not available
```

The selected printer is not available. Select another or check the environment.

Description

Cancels the file currently being printed and clears the rest of the queue.

Prompt

None.

Message

```
The print queue has been cleared
```

Displayed if there were files in the queue.

```
The print queue is empty
```

Displayed if there are no files in the queue.

Print ^PW

Window

Description

Prompts for the number or short name of the window to print, saves that window to a temporary disk file, and adds that file to the print queue.

Prompt

```
Window to print:
```

Enter the number or short name of the window you wish to print.

```
Number of copies?
```

Enter the number of times the window is to be printed.

```
Hold files?
```

Press "Y" to add the window to the queue so that it will not actually print until you release it. Press "N" to make the file available for immediate printing.

```
Which printer?
```

Enter the printer name, for example: lpr.

Message

Printer not available

The selected printer is not available. Select another or check the environment.

Chapter 19: Quit Commands

Using Quit Commands 19-1
Quit Save-Exit 19-2
Quit Terminate 19-4
Quit All Windows 19-6

Chapter 19: Quit Commands

Using Quit Commands

The Quit commands are used to close windows or exit CoEdit. Quit Save-Exit (^QE) will save all modified windows to the disk before quitting. Any UNNAMED windows must be named or the command will be canceled. To quit an UNNAMED window that you do not want saved, use the Window Quit (^QW) command.

The Quit Terminate command will end the program immediately without saving any windows. If there are any modified windows, it will prompt with "Are you sure?" if the SAFETY environment flag is on (see Environment).

Quit ^QE

Save-Exit

Description

Saves all modified windows to the disk and exits CoEdit, returning to the operating system.

Prompt

```
Saving window number nnnn
Enter file name:
```

All UNNAMED windows that have been changed must be assigned a name before being saved to the disk. If a name is not entered for any UNNAMED window, the command is canceled.

```
More files to edit
Are you sure?
```

The File Open or File Read commands have more files awaiting editing. Enter "N" to abort the Quit Exit command, or "Y" to continue with Quit Save-Exit.

Message

```
Print queue is not empty
```

There are files in the print queue. CoEdit cannot end before the queue is empty.

Subwindows are open

This command cannot be used if there are find-all or narrow windows open. Use the Window List command to identify active subwindows, and use the Locate Undo and Block Widen commands as appropriate to complete the editing of those windows.

Restriction

Find All, Defining a Macro, Block Narrow

See Also

Environment option SAFETY

Quit ^QT

Terminate

Description

Exits CoEdit without saving the contents of any windows. All unsaved work will be lost.

Prompt

```
Window(s) have been changed
Are you sure?
```

If any windows have been changed, CoEdit prompts to verify that you want to abandon them.

```
More files to edit
Are you sure?
```

The File Open or File Read commands have more files awaiting editing. Press "N" to abort the Quit Terminate command.

```
Print queue is not empty
```

There are files in the print queue. CoEdit cannot end before the queue is empty. Use the Print Terminate to clear the print queue if desired.

See Also

Environment option SAFETY

Restriction

Defining a Macro

Quit ^QW

All Windows

Description

Discards all open windows whether changed or not and creates a new UNNAMED window.

Prompt

```
Window(s) have been changed
Are you sure?
```

If any windows have been changed, CoEdit prompts to verify that you want to abandon them, but only if the environment option SAFETY is on.

Message

None.

See Also

Environment option SAFETY

Chapter 20: System Commands

Using System Commands 20-1
System Copy 20-2
System Delete 20-3
System Goto 20-4
System List 20-6
System Rename 20-8
System Type 20-9

Chapter 20: System Commands

Using System Commands

The system commands duplicate but improve upon the important operating system directory management commands such as ls, rm, mv, cat, and cp. Including them in CoEdit saves the time of executing them under a shell (see Xternal Commands).

All System commands that require file name input will perform ^F processing during input. Pressing ^F at the System command prompts allows you to view and select the files in one or more directories on the screen.

Description

Makes copies of disk files. You may use file patterns and full path names with drive designations and directories. If the target file exists, it is deleted without warning before the copy begins. This command is equivalent to the operating system cp command.

Prompt

```
File to copy from:
```

Enter the path name and file pattern for files to be copied. ^F processing is available for viewing files and directories.

```
File to copy to:
```

Enter target path and file pattern. ^F processing is available.

Message

```
File does not exist
```

Displayed if no files match the pattern entered at the first prompt above.

Description

Deletes files from the disk. This command is equivalent to the operating system rm command.

Prompt

```
Name of file to delete:
```

Enter the path name and file pattern for files to be deleted. ^F processing is available for viewing directories.

Prompt

```
Are you sure?
```

CoEdit verifies that you wish to delete the named file(s).

```
File does not exist
```

Displayed if no files match the pattern input at the first prompt above.

System ^SG

Goto

Description

Accesses a new working directory for CoEdit. All file names input to CoEdit are assumed to be in the working directory if they do not have a path name. This command allows you to move from one directory to another during an editing session. It does not create a new directory; it only changes the current directory. It is equivalent to the operating system chdir command.

Each time the System Goto command is executed, CoEdit searches the new working directory for a configuration file named co.cfg and a key file named co.nfy. See Chapter 10 for information on these files and what they contain. See Environment Commands for more information on the configurable options. This allows you to create an environment for each directory in which you do your editing. In addition, CoEdit will also search for macro files (extension .cmf by default) in the new directory.

The System Goto command is not an operating system command. It does, however, set the current working directory for CoEdit and the operating system. Any file name given to CoEdit that does not have a path is assumed to reside in the current working directory. If the Environment Configure option N, Find File, is On, then CoEdit will search for the file in the manner described in Appendix G.

Prompt

```
Enter new directory:
```

Enter the directory name for the new working directory. Pressing ^F will only list directories at this prompt.

Message

```
No such directory
```

Displayed if the directory named above could not be found.

System ^SL

List

Description

Displays full or partial listing of the files and subdirectories in one or more named directories. You can display all files and subdirectories that match a given file pattern. Total disk space and available disk space are displayed, along with the number of files matched by the input pattern. Files shown in protected mode (low-intensity) are read-only, while files shown with an asterisk (*) after the file name are hidden. This is also true for directory listings shown with ^F during file input prompts.

Prompt

```
Directory:
```

Enter the path and file pattern to be displayed. The default is all files in the current working directory. Pressing ^F will list only directories at this prompt. Press ABORT to terminate directory input.

```
Press PageUp, PageDown, or ABORT keys:
```

If more than one page of files can be viewed, the PageUp and PageDown keys will scroll the display. Press ABORT to return to the first prompt and enter a new directory to list.

```
Press ABORT to continue:
```

If only one page of files matches the file input pattern, pressing ABORT will return you to the first prompt to enter a new directory to list.

Message

None.

Description

Renames one or more disk files. You can use file patterns and full path names with drive designations and directories. A rename is not possible if a file with the new name already exists. Note that it is possible to move a file from one directory to another with this command.

Prompt

```
File to rename:
```

Enter the path name and file pattern for files to be renamed. ^F processing is available for viewing directories.

```
New file name:
```

Enter new path and file pattern. Both the old and new file patterns must match: if the old pattern includes wildcard characters, the new pattern must also and vice versa. ^F processing is available.

Message

```
File does not exist
```

Displayed if no files match the pattern input at the first prompt above.

Description

Displays the contents of one or more files on the screen. You can view the entire file or lines within a specified range. The display will pause after each screen.

Prompt

```
File to type:
```

Enter path and file pattern of file(s) to type. ^F processing is available to view directories.

```
Starting line:
```

Enter the line number of the first line to be displayed or "T" for top of file.

```
Ending line:
```

Enter the line number of the last line to be displayed or "B" for bottom of file.

```
Press PgDn, End or ABORT keys
```

Press PgDn to move through the current file on display. Press End to end this file and display the next matching file (if any). Press ABORT to terminate the command.

Message

```
File does not exist
```

Displayed if no files match the pattern input at the first prompt above.

```
File name: NAME.EXT
```

Displayed at the bottom of the screen with the current file name. After the file name an A (meaning the file is ASCII) will be displayed.

```
End of file
```

Displayed at the bottom of the screen when the end of the current file is being displayed.

Chapter 21: Text Commands

Using Text Commands 21-1
- Description of Tabs in CoEdit 21-1
- Soft Tabs 21-1
  - In Overtype Mode 21-2
  - In Insert Mode 21-2
- Hard Tabs 21-2
  - In Overtype Mode 21-3
  - In Insert Mode 21-3
- Fixed Tabs 21-3
  - In Overtype Mode 21-3
  - In Insert Mode 21-3

Text Tab Guide 21-4
Text Center Line 21-5
Text Date 21-6
Text Dup Line 21-7
Text Dup One Character 21-8
Text Quote 21-9
Text Remove Marker 21-10
Text Set Marker 21-11

Chapter 21: Text Commands

Using Text Commands

The Text commands in CoEdit offer text formatting and other related functions.

Each window has 10 text markers, numbered 0 through 9, which, if set, appear in the text as reverse-video digits. Text markers are very useful as temporary "place holders" in a file. For example, if you are working on one part of a file and want to quickly scan another part, set a marker at the current location and it will be easy to return there when you are done. Text markers, like block markers, can be deleted by the Delete commands; CoEdit will recognize that they are no longer present in the file.

Neither text nor block markers are saved with your file when it is written to the disk; thus they are not maintained across CoEdit editing sessions.

Description of Tabs in CoEdit

There are three kinds of tabs in CoEdit: soft tabs, hard tabs, and fixed tabs. This section will discuss all three types in detail.

Soft Tabs

The first method of "tabbing" in CoEdit is soft tabs. Soft tabs have stops every eight columns. (The width of the tabs may be set with the Environment Tab and Window Tab commands. See Chapter 10 concerning Environment commands and Chapter 23 concerning Window commands for more information.)

For example, if you type

"this<TAB>is<TAB>a test"

on a new text line, the word "is" will appear at position nine. The word "a" will appear at position seventeen. If you moved to the beginning of the line and typed "1) " (while in insert mode), the word "this" would shift over to the right but the word "is" would remain at position nine. This is done to keep words aligned at tab stops.

If you then go to the beginning of the line and typed "1-", there would be more than eight characters before the "is" and it would no longer remain at position nine. The word "is" would move to seventeen and "a" would move to 25. In other words, a soft tab may vary in size from one to eight (default) and will force the following text to start at the next tab stop.

If you move the cursor to the word "a" and press <BACKSPACE>, the TAB character will be deleted and the text "a test" will move backward to immediately after the word "is". Thus soft tab characters are really maintained by CoEdit; in some word processors pressing <TAB> simply fills the line with ordinary spaces.

In Overtype Mode

The <TAB> key moves the cursor to the next tab stop. Text between the cursor position and the tab stop, if any, is left unchanged. If the tab stop is beyond the end of the current line, a TAB character is added to the end of the line, and the cursor is placed after it.

In Insert Mode

The <TAB> key shifts all characters from the cursor to the end of the current line (if any) to the next fixed tab stop by inserting a TAB character. The cursor is placed just after the TAB character.

Hard Tabs

The second type of tab in CoEdit is a hard tab. When the <TAB> key is struck, it behaves just as a soft tab would. The cursor will go to the next tab stop. If the tab width is the default eight (see the Environment Tab and Window Tab commands), then the tab stops are 1, 9, 17, 25, 33, 41, 49, and so on. Unlike soft tabs, a special tab character is not inserted into the text. Instead, the requisite number of hard spaces are inserted (if appropriate).

In Overtype Mode

The <TAB> key moves the cursor to the next tab stop. Text between the cursor position and the tab stop, if any, is left unchanged. If the tab stop is beyond the end of the current line, spaces are added to the end of the line to the next tab stop, and the cursor is placed after them.

In Insert Mode

The <TAB> key shifts all characters from the cursor to the end of the current line (if any) to the next fixed tab stop by inserting spaces. The cursor is placed just after the spaces.

Fixed Tabs

The last type of tab in CoEdit is the fixed tab. This type of tab most closely simulates the behavior of a standard typewriter. When the <TAB> key is struck, the cursor moves to the next fixed tab stop. The fixed tab stops are like hard tabs but can be manually set at any column position, as opposed to the even spacing (1, 9, 17, and so on) of hard tabs.

In Overtype Mode

The <TAB> key moves the cursor to the next fixed tab stop. Text between the cursor position and the tab stop, if any, is left unchanged. If the tab stop is beyond the end of the current line, spaces are added to the end of the line, and the cursor is placed after them.

In Insert Mode

The <TAB> key shifts all characters from the cursor to the end of the current line (if any) to the next fixed tab stop by inserting a TAB character. The cursor is placed just after the TAB character.

Text ^TT

Tab Guide

Description

Displays a tab guide with tab stop indicators in the current window. Press any key to erase the guide and continue editing. CoEdit will act upon that next keystroke. For example, pressing Ctrl-H will display the help menu.

Prompt

None.

Message

None.

Description

Centers the current line within the current window's margins. Leading and trailing spaces and tabs are ignored and stripped from the line for centering purposes.

Prompt

None.

Message

```
Line would be too long
```

The length of the current line is too long to be centered within the margins set for the current window.

Restriction

Read Only

Description

Inserts the current system date and time into the window at the current cursor position. This is very useful for maintaining comments on the modifications to a source file or routine.

Prompt

None.

Message

```
Line would be too long
```

Since CoEdit's line length limit is 254 characters, the current line cannot accommodate the date string.

Restriction

Read Only

Description

Duplicates the line immediately above from the current cursor position on to the end of the line.

If the cursor is on the first line of the window, or if the previous line ends before the current cursor position, no action is taken. If CoEdit is in Insert mode, the remaining portion of the line above will be inserted into the current line at the current cursor position. If CoEdit is in Overtype mode, the new characters from the line above are overlaid on the current line, replacing its current contents. The cursor moves down to the next line, maintaining the cursor column if possible; if not, it moves to the end of the next line.

Prompt

None.

Message

```
Line would be too long
```

This message appears only in Insert mode, indicating that the resulting line would be longer than 254 characters.

Restriction

Read Only

Text ^TO

Dup One Character

Description

Duplicates the character immediately above the cursor.

If the cursor is on the first line of the window, or if the previous line ends before the current cursor position, no action is taken. If CoEdit is in Insert mode, the character from the line above will be inserted at the current cursor position. If CoEdit is in Overtype mode, the character will replace the one at the current cursor position.

Prompt

None.

Message

None.

Restriction

Read Only

Description

Enters the next character from the keyboard into the text. This allows the entry of control characters into the window. All characters from hex 0 to hex FF are valid.

Prompt

None.

Message

None.

Restriction

Read Only

Text ^TR

Remove Marker

Description

Removes a specified marker from the text without going to it, if it has been set. A marker may also be deleted with any deletion command, including Delete One (the Del key) and Del Previous (the Backspace key).

Prompt

```
Delete which marker (nnn set)?
```

Enter a single digit 0-9 of the marker to remove. Non-digits are ignored. *nnn* indicates which markers are set.

Message

```
Marker not set
```

Displayed if the selected marker has not been set with the Goto Set Marker command.

See Also

The Text Set Marker section for information on setting text markers and the Goto Text Marker section for information on moving the cursor to a text marker.

Description

Sets a text marker at the current cursor location.

Enter the marker number, 0 through 9, and the highlighted digit appears at the cursor. The cursor moves one character to the right.

If the selected marker had been previously set, it is removed from its old location. Markers that are set in blocks of yanked text (see Chapter 25 Yank commands) are removed from the yanked text; thus they may not be moved from one place in the window to another in this manner.

Prompt

```
Set which marker (nnn already set)?
```

Enter a single digit 0-9 of the marker to set. Non-digits are ignored. *nnn* indicates which markers are already set.

Message

None.

See Also

The Text Remove Marker section for information on deleting text markers from the window and the Goto Text Marker section for information on moving the cursor to a text marker.

Chapter 22: Verify Commands

Using Verify Commands 22-1
Verify Compile 22-3
Verify File Reset 22-5
Verify Line Reset 22-6
Verify Next Error 22-7
Verify Previous Error 22-8
Verify Read Error 22-9
Verify Syntax 22-11
Verify Tag 22-13

Chapter 22: Verify Commands

Using Verify Commands

The Verify commands perform syntax checking and compilation support for programmers. Verify Syntax performs a syntax check, looking for improperly matched parentheses, brackets, and braces. The other commands allow you to compile the contents of the current window and highlight lines that generated compiler errors. The Verify Tag command reads a formatted file and highlights one or more lines in the current window.

CoEdit determines the language of the file you are editing and therefore the compiler command it should use to compile your source, by the extension of the file in the current window. Below is a table of the supported file extensions and the languages they represent.

EXTENSION	LANGUAGE
bas	BASIC
c	C
cob	COBOL
for	FORTRAN
pas	Pascal
pl1	PL/I

These extensions can be in either uppercase or lowercase.

The compiler instruction that is passed to operating system is set with the Environment Language command. The file name of the current window is inserted into this instruction and sent to the command processor. (See the Environment Language command in Chapter 10 for more detailed information.) As such, the command should probably be a batch file or shell script. The output of the compiler should be redirected to a file with the same name as the source file and the extension you configure for that language with the Environment Language command (see Chapter 10). The directory where such error files can be found is set with the Environment Configure (option O) command. The compiler command is set with the Environment Language command.

When CoEdit reads an error file, it flags those lines that generated errors and displays them on the screen, highlighted. The Verify Line Reset and Verify File Reset commands turn off the error flag of one or all lines in the current window. The Verify Next and Previous commands move the cursor to the next or previous flagged line.

Some languages are simply translated into other languages with a preprocessor. The code generated by the preprocessor is then sent to the compiler of the new language. Three of the languages listed above are preprocessed.

Description

Saves the contents of the current window if they have been modified and executes the compiler command. When compilation is complete, CoEdit reads the error file generated by the compiler and flags any lines that generated errors. The compilation is executed by appending the file name of the current window to the compiler instruction string and executing the operating system command processor with the resulting string.

Here is an example of a script you might use to compile your source.

Example

```
:
rm -f obj/$1.o
cc $2 -c $1.c 1> err/$1.err 2>&1
rtn=$?
if [ $rtn -ne 0 ]
then
   mv -f $1.o obj/$1.o
fi
exit $rtn
```

Prompt

```
Save to which file:
```

If the current window is UNNAMED, you must supply a file name before the window's contents can be saved.

Message

`Compilation successful`

Displayed if the error file read after the compile listed no errors.

`Error in compilation`

Displayed if the compilation generated errors. Lines that caused errors are flagged and highlighted.

`Language not defined`

The language (as defined by the extension of the file you are editing) is not known to CoEdit. If the window is named UNNAMED, then you must save the contents of the window to the disk with a valid name and extension before you may use the Verify Compile command.

Restriction

Find All, Read Only, Block Narrow

Description

Resets all lines flagged by the Verify Compile or Verify Read commands as having generated errors.

Prompt

None.

Message

None.

Restriction

Find All, Block Narrow

Verify ^VL

Line Reset

Description

Resets the error flag for the current line. If the current line is not flagged, the command has no effect.

Prompt

None.

Message

None.

Restriction

Find All

Description

Moves the cursor to the next line that has been flagged by a Verify Read or Verify Compile command. If there is not another flagged line further down in the file, CoEdit beeps.

Prompt

None.

Message

None.

Restriction

Find All

Verify ^VP

Previous Error

Description

Moves the cursor to the previous line that has been flagged by a Verify Read or Verify Compile command. If there is not another flagged line above in the file, CoEdit beeps.

Prompt

None.

Message

None.

Restriction

Find All

Description

Reads the error file for the source file in the current window and flags all lines that generated errors in the last compilation. The error file is the source file name with extension .err (set with the Environment Language for the language of the current source file) and is searched for in the error directory set by the Environment Language command, option Error Directory. The file in the current window need not have been compiled during the current editing session in order to use this command.

Prompt

None.

Message

```
Cannot open error file
```

The error file for the file in the current window could not be found.

```
Compilation successful
```

Displayed if the last compilation had no errors or warnings. No lines are flagged.

```
Error in compilation
```

Displayed if the last compilation generated errors. Lines that created errors are flagged and highlighted.

Restriction

Find All, Block Narrow

Description

Checks the syntax of the current source file, looking for unmatched and illegal single and double quotes, parentheses, brackets, and braces. The actual syntax check depends on the language of the source file. Comments are excluded.

Prompt

None.

Message

`Syntax check complete`

No errors were found in the current file.

`Improper` *s*

Where *s* is one or more of), }, or] that was found in the file with no corresponding (, {, or [. The error line is highlighted.

Message

`Unmatched` *sss*

One or more of (, {, or [remains unmatched at the end of the source file. CoEdit will display the unmatched symbols. For example, `Unmatched {{{(([` means that one], two), and three } are missing.

`Odd "`

There is an unclosed double-quote in your source.

`Odd '`

There is an unclosed single-quote in your source.

`Unterminated comment`

A comment block in your source was opened but never closed.

`Unopened comment terminated`

A comment block in your source was closed but never opened.

Restriction

Find All

Description

This command prompts for the name of a tag file to read. The file name you supply is assumed to be appropriate for the language of the source in the current window. CoEdit will then proceed as with the Verify Read Error command described previously.

If the language of the source file in the current window is unknown to CoEdit, then CoEdit will read the error file, looking for lines in the following format:

filename.ext nnnn message

where *filename.ext* is the name of the file in the current window; *nnnn* is the line number to be flagged; and *message* is a message terminated by a newline or an end-of-file. For example,

```
compute.bpm 183 Punctuation error
```

would indicate to CoEdit that line 183 of the current window (presuming it contains the file compute.bpm) is to be tagged, and the message "Punctuation error" is to be attacted to it. There may be up to 255 tagged lines. Duplicate tagging of lines is ignored.

This feature of CoEdit is very powerful and opens up a range of possibilities for integrating your own tools with CoEdit, for example, a lint program (one which checks program correctness). Even if the lint program does not provide messages in the above format, you could easily write a filter program to massage it. This feature can be applied to any number of useful programs, even ones you write yourself. You will then have a tight integration between your program and CoEdit.

Prompt

```
Read from which file?
```

Enter the name of the file containing the tagging information.

Message

```
File does not exist
```

The file you have entered does not exist. Enter another file to read.

Restriction

Find All, Block Narrow

Chapter 23: Window Commands

Using Window Commands 23-1
Windows on the Screen 23-1
    Opening and Closing Windows 23-2
    Moving Between Windows 23-2
    Window Attributes 23-2
    The Window Commands 23-3
Window Assign 23-5
Window Bury 23-6
Window Compare 23-7
Window Copy 23-9
Window Expand 23-11
Window Hide 23-12
Window List 23-13
Window Move 23-14
Window Next 23-16
Window Open 23-17
Window Previous 23-18
Window Quit 23-19
Window Read-Only 23-20
Window Size 23-21
Window Tab 23-22
Window Undo 23-23
Window Visible 23-25
Window Slide 23-26

Chapter 23: Window Commands

Using Window Commands

The window is the most important element of CoEdit's operation. More than a screen area, a CoEdit window is actually an environment in which editing takes place. Each window is completely independent from all others and may have its own characteristics.

CoEdit maintains a list of windows in the order in which they were created and assigns a unique number to each window. No matter how many windows are open, there is only one current window, the window with the highlighted border. The cursor is always located in this window, and most commands affect only its contents. There is always a current window; if all open windows are closed, CoEdit will create a new window automatically. There is no "opening menu" as in some word processing programs.

The number of open windows is displayed on the status line, which also reflects the state of the current window; it displays the file name for this window, the current line and column positions, and the total number of lines in the window. For information on other open windows, use the Window List command.

Windows on the Screen

CoEdit can display a virtually unlimited number of windows on the screen. You may manually hide and unhide windows as well as resize them to control the number of windows displayed.

CoEdit allows windows to be expanded. Often, with more than one window open, the windows do not occupy the entire screen. However, if a window is expanded, it will take up the full screen whenever it is the current window.

Opening and Closing Windows

Windows can be opened in four ways. First, CoEdit will open a window automatically if the number of open windows reaches zero. Second, the File commands that load one or more files into CoEdit's memory will create new windows and make one the current window. Third, the Window Open command will create a new window and make it the current window. Fourth, Macro Uncompile will open a new window and load the source to a macro into it.

Windows can be closed in a number of ways. The Window Quit, File Done, File Update, Quit Save-Exit, and Quit All Windows commands all close one or more windows. The first two affect only the current window; the others affect all or all modified windows. If any of these commands closes the last open window, CoEdit will automatically create a new one and make it the current window.

Moving Between Windows

When more than one window is open, the Window Next, Window Previous, and Goto Window commands can be used to change the current window. The Window Open and File Open commands always make the window they create current, and as mentioned above, there are commands that close one or more windows and change the current window. If the current window is changed, the screen may be redrawn if either the previous or the new current window is expanded.

Window Attributes

In addition to the display attributes mentioned above (expanded and hidden), windows have three other major attributes: read-only, ASCII, and modified. The read-only attribute is set ON when CoEdit detects that the file loaded into the window is stored on the disk read-only (see your operating system manual) or if you load a file that is already in CoEdit's memory. This attribute can also be set manually with the Window Read/Only command. The ASCII flag indicates how the file was read from the disk and how it will be saved. This also may be toggled manually. The modified attribute is set by CoEdit and reflects whether or not the contents of the window have been changed since the last save or load. It can be reset with the Miscellaneous Changed command. For a complete list of open windows with their attributes, use the Window List command.

The Window Assign command can be used to assign a short name to the current window. This short name can then be used at any prompt requesting the identification of a window (for example, to copy from, to unhide, to go to). This feature provides a shorthand way of remembering window contents during editing sessions that involve many open windows and is particularly useful in macros for "marking" windows so direct movement between the windows is possible. The short name can be from one to four characters long and should not be a number. If a window has a short name, it is displayed on the border of the window between the window's number and the window's file name.

The Window Commands

The Window commands fall generally into four groups: commands for changing the current window, changing current window attributes, changing the window's appearance, and transferring text between windows.

Changing the current window:

```
^WN   - Make next window in list current
^WO   - Open a new window and make it current
^WP   - Make previous window in list current
^WQ   - Quit the current window; the last window
        becomes the current
^GW   - Goto window (See Chapter 12)
```

Changing window attributes:

```
^WA   - Assign a 1-4 char name to this window
^WF   - Change the ASCII status for this
        window
^WR   - Toggle the read-only flag for this
        window
^WT   - Change tabs for this window
```

Changing the window's appearance:

```
^WB    - Buries the current window
^WE    - Toggle the expanded flag for this window
^WH    - Hide the current window
^WS    - Size or resize a window on the screen
^WU    - Unhide a hidden window
^WW    - Move a window on the screen
```

Transferring text between windows:

```
^WC    - Copy a block from another window
^WM    - Move a block from another window
```

Other:

```
^WD    - Compares the text in two windows
^WL    - List all windows with their attributes
```

Description

Allows a one to four character short name to be assigned to the current window. This short name can then be used to identify the window at all prompts that require a window to be identified. Short names are particularly useful in macros for "marking" windows so direct movement between the windows is possible. It is also useful for giving unique names to windows when editing files with the same name.

Prompt

```
Name to assign to this window:
```

Enter a 1-4 character short name for this window. If you press ENTER alone the short name for this window will be deleted.

Message

None.

Window ^WB

Bury

Description

"Buries" the current window. The current window will be placed first in the screen redraw cycle. This means that it will be on the bottom of all other open windows. This is a way to quickly move a window out of vision's way. Note that this command has no effect when there is only one window displayed on the screen, or if it does not intersect with other windows.

Prompt

None.

Message

None.

See Also

Window Hide for information on removing a window from the screen entirely.

Description

Compares the text of the current window with the text of another window. This command will position the windows' cursors at the positions where they differ. If the text is the same, the cursors remain unmoved. This command is extremely useful for differentiating between versions of the same file.

The comparison may be performed while ignoring case and ignoring white space. In addition, the comparison can begin at the current position or at the beginning of file.

Prompt

```
Window to compare against:
```

Enter the number or short name of the window against which the text of the current window should be compared.

```
Enter compare options:
```

Enter the desired options for the comparison. B starts the comparison at the beginning of file. H starts the comparison here, at the current cursor position. I ignores case, and C considers case. Finally, M means white space matters, and D means white space does not matter.

Message

```
Windows compare OK
```

The windows are the same, given the compare options you chose.

```
The windows differ here
```

The windows differ, given the compare options you chose, at the current cursor position.

Restriction

None.

Description

Copies a block defined in another window into the current window, inserting the text of the block at the cursor location. The source window is not changed.

Prompt

```
Copy block from which window:
```

Enter the number or short name of the window whose block is to be copied to the current window. The windows with blocks defined are listed on the screen.

```
Press the PageUp, PageDown, or ABORT keys:
```

If all windows with blocks defined cannot be listed on one screen, use the PageUp and PageDown keys to scroll through the list. ABORT cancels the command.

Message

```
No window has a block defined
```

Displayed if none of the windows currently open has a correctly defined block.

Restriction

Find All, Read Only

Description

Toggles the expanded attribute for the current window. If a window is expanded, it will be displayed on the whole screen when it is the current window. Each execution of this command toggles the value of the attribute between YES and NO.

Prompt

None.

Message

None.

See Also

Environment options for information on whether or not new windows will be expanded automatically.

Window ^WH

Hide

Description

Hides the current window and makes it inaccessible to the Window Next, Window Previous, Window Copy, Window Move, and Window Undo commands. Manually hidden windows can be unhidden by the Window Visible command, or by using the Goto Window command. Use Window Hide to temporarily put a window out of sight. Hidden windows are processed by those functions that save all modified windows; they simply never appear on the screen.

If there are no more visible (non-hidden) windows open, CoEdit automatically opens a new UNNAMED window and makes it the current window.

Prompt

None.

Message

None.

Description

Displays a list of all open windows with their numbers, attributes, short names, and associated file names. The attributes are as follows:

Attribute	Meaning
C	Changed - window has been changed
H	Hidden - window is hidden
E	Expanded - window is expanded
R	Read-Only - window is read-only
N	Narrow - window is narrow (see Block Narrow)
F	Find all - window is in Find All mode

Prompt

```
Press the PageUp, PageDown, or ABORT keys:
```

If all open windows cannot be listed on one screen, use the PageUp and PageDown keys to scroll through the list. ABORT terminates the command.

Message

None.

Window ^WM

Move

Description

Moves a block defined in another window into the current window, inserting the text of the block at the cursor location. The block is deleted from the source window.

Prompt

```
Move block from which window:
```

Enter the number or short name of the window whose block is to be moved to the current window. The windows with blocks defined are listed on the screen.

```
Press the PageUp, PageDown, or ABORT keys:
```

If all windows with blocks defined cannot be listed on one screen, use the PageUp and PageDown keys to scroll through the list. ABORT cancels the command.

Message

```
No window has a block defined
```

Displayed if none of the windows currently open has a correctly defined block.

Restriction

Find All, Read Only

Window ^WN

Next

Description

Makes the next unhidden window the current window. If there is no other unhidden window in the window list, the command has no effect. If the expanded attribute of the new current window differs from that of the old current window, or the window is not completely visible, the screen will be redisplayed.

Prompt

None.

Message

None.

Description

Opens a new window and makes it the current window. The window is given the file name UNNAMED and is initialized to be empty.

Prompt

None.

Message

None.

See Also

Environment options for information on setting whether or not new windows will be expanded automatically.

Window ^WP

Previous

Description

Makes the previous unhidden window in the window list the current window. If there is no other unhidden window in the window list, the command has no effect. If the expanded attribute of the new current window differs from that of the old current window, or the window is not completely visible, the screen will be redisplayed.

Prompt

None.

Message

None.

Description

Abandons the contents of the current window and makes the previous unhidden window in the window list the current window. If there is no other unhidden window in the window list, CoEdit automatically creates a new window and makes it current.

Caution

```
File has been altered. OK to quit?
```

If the changed attribute of the current window is set and the environment variable SAFETY is set, then CoEdit verifies that they should be abandoned.

See Also

Environment options for information on controlling whether or not warnings are displayed.

Restriction

Find All, Block Narrow

Window ^WR

Read-Only

Description

Toggles the read-only attribute for the current window. If a window is read-only, then no modifications may be made to it at all and many commands are restricted. Characters cannot be deleted, added, or changed. Lines cannot be moved, added, or deleted. Each execution of this command toggles the value of the attribute between YES and NO. Files that are stored on the disk as read-only and loaded into CoEdit have this attribute permanently set to YES. The read-only attribute of windows containing duplicate files in CoEdit may be reset.

Prompt

None.

Message

None.

Description

Sizes or resizes the current window. The h, j, k, and l keys make the window larger by moving the border pointed to by the key you hit out one space from the center of the window. (For example, striking the l key moves the right border one column right. If the border you desired to move is at the screen border, the window will grow in the opposite direction (resulting in the same expansion of the window).

To reduce the size of the window, use Shift plus the same H, J, K, and L keys. For example, striking J will pull the top border of the window down one line; L will pull the left border in one line.

Hitting ENTER, ABORT, or Ctrl-C will end the command. Note that resizing expanded windows makes them unexpanded.

Prompt

```
Sizing
```

This message will appear blinking on the command line until the command is terminated.

Message

None.

Window ^WT

Tab

Description

Sets the tab configuration for the current window only. See Environment Tab for detailed instructions on how this command works.

Prompt

```
Enter a letter to select an item:
```

Enter a D, T, W, S, J, L, or R to select an item. Press ENTER to accept the changes, or either ABORT or ^C to abort the changes.

Message

```
Number too low - range is 1 to 16
```

The number you entered for the tab width is too small. Select a number in the given range.

```
Number too high - range is 1 to 16
```

The number you entered for the tab width is too large. Select a number in the given range.

See Also

Environment Tab for information on setting the tabs for all new windows.

Description

Allows you to bring back into the current window text that has been deleted from another window.

Prompt

```
Undo from which window?
```

Enter the number or short name of the window whose deleted text you want to undo.

```
Deletion to undo?
```

Enter the number of the deleted buffer you want to insert at the cursor.

Message

```
No windows with delete buffers
```

No open windows have any delete buffer to undo.

See Also

Delete Undo for information on the operation of the undo function.

Chapter 7 concerning Block Commands for information on inserting text into the window.

Restriction

Find All, Read Only

Description

Allows the hidden attribute of a window to be changed from YES to NO. This will make the hidden window visible once again. Unhiding a window does not force its immediate screen display; it simply permits it to be displayed.

Prompt

```
Window to unhide:
```

Enter the number or short name of the window to unhide.

```
Press the PageUp, PageDown, or ABORT keys:
```

If all hidden windows cannot be listed on one screen, use the Pg keys to scroll through the listing. ABORT will cancel the command.

Message

```
No windows are hidden
```

Displayed if there are no hidden windows.

See Also

Window Hide for information on hiding windows.

Window ^WW

Slide

Description

Slides a window around on the screen. Use the h, j, k, or l (left, down, up, and right, respectively) to move the window. Hitting ENTER, ABORT, or Ctrl-C will end the command. Note that you may not move a window that is expanded or is actually the size of the screen.

Prompt

None.

Message

```
Sliding
```

Displayed flashing on the status line while you are moving the window.

Chapter 24: Xternal Commands

Using Xternal Commands 24-1
Xternal Chmod 24-2
Xternal Execute 24-3
Xternal Make Directory 24-4
Xternal Remove Directory 24-5
Xternal Shell 24-6
Xternal Touch 24-7

Chapter 24: Xternal Commands

Using Xternal Commands

The Xternal commands perform a variety of useful disk management tasks that include some DOS-like and some UNIX-like functions. Make Dir and Remove Dir create and delete directories in the same way as their operating system counterparts.

The Xternal Execute command will pass your input string to a subshell. To create a subshell and execute multiple commands, use the Xternal Shell command. This command simply loads the operating system command processor. To leave the command processor, type EXIT.

The Xternal Attribute command sets a program-wide attribute flag for the type of files to be included in directory searches. The attribute can be changed to include only normal files or hidden, read-only, and archived files as well.

Xternal ^XC

Chmod

Description

Changes the attribute of one or more files on the disk. You can add a characteristic (such as read/write permission) to any set of files or replace their attribute with a new set of characteristics. See your operating system manual for a discussion of file attributes.

Prompt

```
Name of file to chmod:
```

Enter the path and file pattern of the files whose attributes are to be changed.

```
New file mode:
```

Enter the permissions just as you would for the UNIX chmod command. For example, 777, +w, go+r, and -x are all valid responses. (See your UNIX documentation for the chmod command.)

Message

```
File does not exist
```

Displayed if there are no files that match the file pattern input at the first prompt.

Description

Invokes the operating system command processor to execute an input string as if it had been typed at the system prompt.

Prompt

```
Enter command:
```

Enter the string to be passed to the command processor for execution.

Message

```
Could not execute command
```

There was insufficient memory available to execute the command processor. Free some memory in CoEdit and retry the operation.

Xternal ^XM

Make Directory

Description

Creates a new disk directory anywhere in the directory hierarchy. See your operating system manual for a discussion of directories and sub-directory structures.

Prompt

```
Name of directory to make:
```

Enter the name of the new directory. Pressing ^F at this prompt will display all directories matching the path and pattern in the input field.

Message

```
File exists
```

Displayed if the directory name input above already exists as a file.

```
Cannot make directory
```

Displayed if the directory name input above already exists as a directory.

Description

Deletes a disk directory anywhere in the directory hierarchy. The directory must be empty before it can be removed.

Prompt

```
Name of directory to remove:
```

Enter the name of the directory to be deleted. Pressing ^F at this prompt will display all directories matching the path and pattern in the input field.

Message

```
Cannot remove that directory
```

Displayed if the named directory is not empty. Directories cannot be deleted unless they contain no files at all.

```
No such directory
```

Displayed when the input name is not a directory or cannot be found.

Xternal ^XS

Shell

Description

Loads a copy of the operating system command processor to create a shell. You may execute one or more programs or commands and return to CoEdit by entering the operating system command EXIT.

Prompt

None.

Message

```
Could not execute command
```

There was insufficient memory available to execute the command processor. Free some memory in CoEdit and retry the operation.

Description

Sets the date and time for one or more disk files to be the current system date and time, without actually loading them and saving the files.

Prompt

```
Name of file to touch:
```

Enter the path and file pattern selecting the files whose date and time should be updated to the current system date and time. ^F processing is available.

Message

```
File does not exist
```

Displayed if there are no files that match the input path and pattern.

```
A file name is required
```

Displayed if the input path and pattern identify a directory and not a file.

Chapter 25: Yank Commands

Using Yank Commands 25-1
Summary of Yank Commands 25-2
Yank To Bottom of File 25-3
Yank To a Character 25-4
Yank Line 25-5
Yank To End of Screen 25-6
Yank To Finish of Line 25-7
Yank To Home 25-8
Yank Word Left 25-9
Yank Word Right 25-10
Yank To Start of Line 25-11
Yank To Top of File 25-12
Yank Word 25-13

Chapter 25: Yank Commands

Using Yank Commands

Commands in the Yank group copy blocks of text from the current window and store them in the list of delete buffers maintained for each window.

The blocks of text to be copied are always determined with reference to the current cursor location. Thus Yank to Start copies all text from the cursor to the start of the current line. Yank to Home of screen, Top of file, etc. work the same way. The command letters in this group are very similar to those in the Cursor commands and are identical to those in the Delete commands. (See Chapter 13 concerning Internal Commands for a description of delete buffers.)

The Misc Undo command will "undo" the last yanked or deleted text by inserting the first entry on the delete buffer stack into the text at the current cursor location. The Delete Undo command displays the delete buffer stack and prompts for the number of the buffer to insert. Note that each window has an independent buffer stack; to yank a line in one window and undo that line in another, see the Window Undo command. (See the Window Copy and Move commands for interwindow text transfer.)

Summary of Yank Commands

The Yank commands can be arranged as follows:

From the cursor backward:

COMMAND	LOCATION	KEY SEQUENCE
TOF	top of file	^YT
Home	home of window	^YH
Start	start of line	^YS
Left Word	begin of word	^YL

From the cursor forward:

COMMAND	LOCATION	KEY SEQUENCE
Right Word	end of word	^YR
Finish	end of line	^YF
End	end of window	^YE
BOF	bottom of file	^YB

Others:

COMMAND	LOCATION	KEY SEQUENCE
To a char	to a character	^YC
Word	the current word	^YW
Line	the current line	^YD

The following sections describe each of these commands in alphabetical order by key sequence.

Note that all Yank commands perform memory allocations; the message **Not enough memory** may appear to signal insufficient memory to perform a yank command. (See Chapter 13 concerning Internal Commands, for tips on conserving memory.)

Description

Yanks all text from the current character to the last character in the file.

Prompt

None.

Message

None.

Yank ^YC

To a Character

Description

Yanks all text from the cursor up to but not including an input character. The character input must be on the current line or the command has no effect.

Prompt

```
To what character?
```

Input the character up to which text should be yanked.

Message

```
Character not found
```

The input character could not be found after the cursor on the current line.

Description

Yanks the entire current line. If the environment option Move on Yank Line is set to On, then the cursor will move down if possible. This way you could yank several lines and undo those lines somewhere else. If the option is Off, then you could yank one line many times.

Prompt

None.

Message

None.

See Also

Environment option Move on Yank Line for a complete description of this option.

Yank ^YE

To End of Screen

Description

Yanks all characters from the cursor to the last character shown in the current window.

Prompt

None.

Message

None.

^YF

Yank

To Finish of Line

Description

Yanks all characters from the cursor to the end of the current line. The command has no effect when the cursor is at the end of the line.

Prompt

None.

Message

None.

Yank ^YH

To Home

Description

Yanks all characters from the cursor to the first character shown in the current window.

Prompt

None.

Message

None.

Description

Yanks from the cursor to the beginning of the current word if the cursor is in the middle of a word.

Yanks all delimiters to the end of the previous word (or the beginning of the line) if the cursor is at the beginning of a word or on a delimiter.

The command has no effect at the beginning of the line.

Prompt

None.

Message

None.

Yank ^YR

Word Right

Description

Yanks the entire word and trailing delimiters if the cursor is at the beginning of a word.

Yanks from the cursor to the end of the current word if the cursor is in the middle of a word.

Yanks all delimiters to the beginning of the next word (or end of line) if cursor is on a delimiter.

The command has no effect at the end of the line.

Prompt

None.

Message

None.

^YS

Yank

To Start of Line

Description

Yanks all characters from the left of the cursor to the start of the line. The command has no effect when the cursor is at the start of the line.

Prompt

None.

Message

None.

Yank ^YT

To Top of File

Description

Yanks all characters from the left of the cursor to the first character of the file.

Prompt

None.

Message

None.

Description

Yanks the entire current word if the cursor is on a non-delimiter; yanks all delimiters forward to the next word (or end of line) if the cursor is on a delimiter.

Prompt

None.

Message

None.

Chapter 26: Miscellaneous Commands

Using Miscellaneous Commands 26-1
Misc Again 26-2
Misc Back Tab 26-3
Misc Changed 26-4
Misc Date 26-5
Misc Evaluate 26-6
Misc Open Above 26-8
Misc Ins/Ovr 26-9
Move Right 26-10
Misc New Line 26-11
Misc Open Below 26-12
Misc Paint Screen 26-13
Misc Repeat 26-14
Misc Sound 26-15
Misc Tab 26-16
Misc Undo Delete 26-17
Misc Execute 26-18
Misc Enter-Auto 26-19
Misc Enter 26-20

Chapter 26: Miscellaneous Commands

Using Miscellaneous Commands

This group of commands includes five commands attached to single keystrokes and the expression evaluator. The Again, Newline, Open Below, Repeat, and Undo Delete commands are assigned to ^A, ^N, ^O, ^R, and ^U, respectively.

The other commands are ones that do not fall easily into the other categories of CoEdit commands, yet are very useful for programming.

Misc ^ZA or ^A

Again

Description

Repeats the last character entered or command executed. No action is taken if Misc Again is the first command executed after CoEdit loads. This command may be used if the last command was a macro as well.

Prompt

None.

Message

None.

Description

Moves the cursor back to the previous tab stop. If there are no more tab stops on the current line, CoEdit will beep.

Prompt

None.

Message

None.

Misc ^ZC

Changed

Description

Resets the changed flag for this window. The changed flag indicates whether or not the contents of this window have been modified. After execution of this command, CoEdit will no longer treat this window as saved. This command is useful if the contents of the window were changed accidentally.

Use caution with this command. If you make a change to a window that you want saved to disk, executing this command will prevent all commands except File Save (see Chapter 11) from saving the window.

Prompt

None.

Message

None.

Restriction

Find All

Description

Displays the current system date and time on the screen. This command has no affect on any windows.

Prompt

```
Press ABORT to continue
```

Press ABORT to resume editing.

Message

None.

See Also

Text Date for inserting the date and time into the text.

Description

Evaluates one or more expressions. Results are displayed in four numeric bases: decimal, hexadecimal, octal, and binary. Expressions may include parentheses. Operators are:

+	add
-	subtract
*	multiply
/	divide
%	modulus
#	power
^	exclusive OR
&	logical AND
|	logical OR

The numbers you enter are decimal by default, though the default may be changed with the Environment Configure command. (See Chapter 10.)

Numbers may be preceded with a base override in the form of 0*t* where *t* is: o or O for octal, b or B for binary, x or X for hexadecimal, or d or D for decimal. Thus the hex number F04D would be entered 0xF04D.

Expressions are evaluated from left to right, with no precedence of operators.

Prompt

```
Enter an expression to be evaluated.
```

Message

```
Unknown operator
```

Displayed if an illegal operator, number, or digit was entered.

Misc

^ZH or Alt-O

Open Above

Description

Opens a new blank line above the current line and places the cursor on it.

If the environment variable AutoIndent is set (see Chapter 10), then the new line will be indented to the same position as the current line. The cursor will be placed at the end of the new line.

Prompt

None.

Message

None.

See Also

Misc Open Below

Restriction

Find All, Read Only

Description

Toggles the Insert/Overtype flag for CoEdit.

Prompt

None.

Message

None.

See Also

Insert/Overtype descriptions in Chapters 2 and 4.

Move ^ZM

Right

Description

Moves the cursor one position right on the current line, whether or not the cursor is at the end of the line.

If the cursor is at the end of the line, then a space is appended to the current line and the cursor moves right one position.

Otherwise, the cursor simply moves right to the next column. This is useful in macros when you need to ensure that the cursor will move to the correct columnar position.

Prompt

None.

Message

None.

See Also

Goto Column for further information and another method of absolute column addressing.

Restriction

Read Only

Description

Inserts an ENTER character at the current cursor position, creating a new line containing all characters to the right of the cursor. Unlike the action performed by pressing ENTER, the cursor remains at its current position.

Prompt

None.

Message

None.

Restriction

Find All, Read Only

Misc ^ZO or ^O

Open Below

Description

Opens a new blank line after the current line and places the cursor on it.

If the environment variable AutoIndent is set (see Chapter 10), then the new line will be indented to the same position as the current line. The cursor will be placed at the end of the new line.

Prompt

None.

Message

None.

See Also

Misc Open Above

Restriction

Find All, Read Only

Description

Repaints the screen.

If the screen becomes inaccurate due to other tasks writing to it, this command will redisplay the correct CoEdit screen.

Prompt

None.

Message

None.

Misc ^ZR or ^R

Repeat

Description

Sets up a repeat count for the next operation. The number entered as the repeat count will determine the number of times the operation is to be performed. For example, 50 PageDown would perform fifty page-downs. The repeat count does not apply to the Repeat command.

Prompt

```
Repeat Count?
```

Enter the number of times the following operation is to be performed.

Message

None.

Description

Sounds a tone on the system speaker. Often used in macros to signal the end of a long operation.

Prompt

None.

Message

None.

See Also

The Environment Configure Command for information on how to change the tone generated.

Misc — ^ZT or the Tab Key

Tab

Description

Places a tab at the current cursor position. For a discussion of the tabs in CoEdit, refer to Chapter 21.

Prompt

None.

Message

None.

Restriction

Read Only

Description

Inserts the last deleted text into the current window at the cursor location. Each window maintains its own list of deleted texts.

Prompt

None.

Message

```
No delete buffers exist
```

Displayed if there are no delete buffers for the current window.

See Also

The Delete Undo command in Chapter 9 for further information.

Restriction

Read Only

Misc ^ZX

Execute

Description

Executes a CoEdit command or macro. This allows you to access commands even if they are not bound to keys on the keyboard. Macros have highest priority; internal CoEdit commands are next.

Note that this command is not repeatable.

If, after the successful execution of a command using Miscellaneous Execute, the Miscellaneous Again function is performed, then the function executed by the Miscellaneous Execute command will be repeated, not the Miscellaneous Execute command itself.

Prompt

```
Enter command:
```

Enter the name of the macro or command you wish to execute. You may use the ^F (file list) feature to display a directory of available macros. The command must be either the command name (for example, Env_configure) or a two-letter command name (EC).

Message

```
Could not execute command
```

The command you requested is not defined.

Description

Performs the operation of the ENTER key, plus AutoIndents to the indented position of the line above. This function performs no differently than if the environment option AutoIndent were on and the normal ENTER key were struck.

Prompt

None.

Message

None.

Restriction

Find All, Read Only

Misc

^ZZ or ENTER

Enter

Description

Performs the Enter operation; that is, it creates a new line while you are in text entry.

If you are in Overtype mode, the cursor is simply brought down to the next line on column 1.

If you are on the last line of the window, then a new line is created at the bottom of the window.

If you are in Insert mode, all the characters from the cursor onwards are brought down into a new line immediately following the current line.

Restriction

Find All, Read Only

Appendix A: CoEdit Commands and Macros

The following are CoEdit's commands. On the left are the keystrokes required to invoke the command. In the middle is the short description of the command. The name on the right should be used to invoke the command in CoEdit CML macros. (For further information, see Chapters 5 and 6.)

BINDING	DESCRIPTION	NAME
BA	Block All	Block_all
BB	Block Begin	Block_begin
BC	Block Copy	Block_copy
BD	Block Delete	Block_delete
BE	Block End	Block_end
BF	Block Filter	Block_filter
BL	Block Lower Case	Block_lower
BM	Block Move	Block_move
BN	Block Narrow	Block_narrow
BP	Block Program	Block_program
BQ	Block Quick	Block_quick
BR	Block Remove	Block_remove
BS	Block Save	Block_write
BT	Block Text Wrap	Block_wrap
BU	Block Upper Case	Block_upper
BV	Block Vertical	Block_vertical
BW	Block Widen	Block_widen
BY	Block Yank	Block_yank
CB	Curs Bot Wind	Go_window_bot
CD	Curs Down Line	Go_down_line
CE	Curs End Wind	Go_window_end
CF	Curs Finish Ln	Go_end_line
CH	Curs Home	Go_home
CL	Curs Left Word	Go_left_word
CM	Curs Mid Wind	Go_mid_screen
CR	Curs Right Wrd	Go_right_word
CS	Curs Start Ln	Go_begin_line
CT	Curs Top Wind	Go_window_top
CU	Curs Up Line	Go_up_line

BINDING	DESCRIPTION	NAME
CW	Curs End Word	Go_end_word
CY	Curs Left One	Go_left
CZ	Curs Right One	Go_right
DB	Del To BOF	Del_eof
DC	Del To a Char	Del_to_char
DD	Del Line	Del_line
DE	Del To EOS	Del_eos
DF	Del To Finish	Del_end_line
DH	Del To Home	Del_bos
DL	Del Word Left	Del_word_left
DO	Del One char	Del_char
DP	Del Prev Char	Del_prev
DR	Del Word Right	Del_word_right
DS	Del To Start	Del_begin_line
DT	Del To TOF	Del_tof
DU	Del Undo	Del_undo
DW	Del Word	Del_word
DY	Del Yank	Del_yank
EA	Env Attribute	Env_attr
EC	Env Configure	Env_configure
EL	Env Language	Env_language
EM	Env Macro	Env_macro
EP	Env Printer	Env_print
ER	Env Read	Env_read
ES	Env Save	Env_save
ET	Env Tab	Env_tabs
EU	Env Update	Env_update
FA	File Another	File_another
FD	File Done	File_done
FE	File Exchange	File_exchange
FL	File Load	File_load
FM	File Merge	File_merge
FN	File Next	File_next
FO	File Open	File_open
FR	File Read	File_read
FS	File Save	File_save
FU	File Update	File_update
GB	Goto Beg Blk	Goto_block_begin
GC	Goto Column	Goto_column
GE	Goto End Blk	Goto_block_end
GI	Goto Ins Mode	Goto_insert_mode

BINDING	DESCRIPTION	NAME
GL	Goto Line	Goto_line
GM	Goto Matching	Goto_matching
GO	Goto Overtype	Goto_overtype
GP	Goto Previous	Goto_previous_pos
GT	Goto Text Mark	Goto_marker
GW	Goto Window	Goto_window
IC	Intrn Char Value	Char_value
ID	Intrn Del Buffer	Del_buffer
IK	Intrn Key Code	Key_code
IS	Intrn Snow Check	Snow_check
IT	Intrn Table	Char_table
JB	Jump Up 1/2 Pg	Page_up_half
JD	Jump Down Line	Jump_down_line
JE	Jump End File	Jump_to_eof
JF	Jump Dn 1/2 Pg	Page_down_half
JH	Jump Home	Jump_window_top
JL	Jump Last	Jump_window_bot
JN	Jump Next Page	Page_down
JP	Jump Prev Page	Page_up
JS	Jump Scroll Lck	Scroll_lock
JT	Jump Top File	Jump_to_bof
JU	Jump Up Line	Jump_up_line
KC	Key Commands	Key_commands
KL	Key Fkey List	Key_list
LA	Loc Again	Locate_again
LM	Loc Marked	Locate_marked_lines
LN	Loc Next	Locate_next
LP	Loc Previous	Locate_previous
LR	Loc Replace	Locate_replace
LS	Loc String/pat	Locate_string
LU	Loc Undo	Locate_undo
MC	Macro Compile	Macro_compile
MD	Macro Define	Macro_define
ME	Macro End Def	Macro_end
MF	Macro Flush	Macro_flush
MG	Macro Go	Macro_go
MK	Macro Key Name	Macro_key_name
ML	Macro List	Macro_list
MS	Macro Save	Macro_save
MT	Macro Trace	Macro_trace
MU	Macro Uncompile	Macro_uncompile

BINDING	DESCRIPTION	NAME
PA	Print Again	Print_again
PB	Print Block	Print_block
PC	Print Cancel	Print_cancel
PD	Print Delete	Print_delete
PE	Print Eject	Print_eject
PF	Print File	Print_file
PH	Print Hold	Print_hold
PQ	Print Queue	Print_queue
PP	Print Pause	Print_pause
PR	Print Release	Print_release
PT	Print Terminate	Print_terminate
PW	Print Window	Print_window
QE	Quit Save-Exit	Quit_exit
QT	Quit Terminate	Quit_terminate
QW	Quit All Winds	Quit_windows
SC	Sys Copy Files	Copy_file
SD	Sys Delete	Delete_file
SG	Sys Change Dir	Change_dir
SL	Sys List Files	List_file
SR	Sys Rename	Rename_file
ST	Sys Type Files	Type_file
TC	Text Center	Text_center
TD	Text Date	Text_date
TL	Text Dup Line	Text_dup_line
TO	Text Dup One	Text_dup_char
TQ	Text Quote	Text_quote
TR	Text Remov Mark	Text_del_mark
TS	Text Set Marker	Text_set_mark
TT	Text Tab Guide	Text_tab_guide
VC	Ver Compile	Compile
VF	Ver File Reset	Reset_file
VL	Ver Line Reset	Reset_line
VN	Ver Next Error	Next_error
VP	Ver Prev Error	Prev_error
VR	Ver Read Error	Read_errors
VS	Ver Syntax	Check_syntax
VT	Ver Tag	Tag_lines
WA	Wind Assign	Window_assign
WB	Wind Bury	Window_bury
WC	Wind Copy	Window_copy
WD	Wind Compare	Window_compare

BINDING	DESCRIPTION	NAME
WE	Wind Expand	Window_expand
WF	Wind Format	Window_format
WH	Wind Hide	Window_hide
WL	Wind List	Window_show
WM	Wind Move	Window_move
WN	Wind Next	Window_next
WO	Wind Open	Window_open
WP	Wind Previous	Window_previous
WQ	Wind Quit	Window_quit
WR	Wind Read-Only	Window_read_only
WS	Wind Size	Window_size
WT	Wind Tabs	Window_tabs
WU	Wind Undo	Window_undo
WV	Wind Visible	Window_unhide
WW	Wind Slide	Window_slide
XA	Xtrnl Attribute	Set_attribute
XC	Xtrnl Chmod	Chmod
XE	Xtrnl Execute	Execute
XM	Xtrnl Make Dir	Make_dir
XR	Xtrnl Remov Dir	Remove_dir
XS	Xtrnl Shell	Shell
XT	Xtrnl Touch	Touch
YB	Yank To BOF	Yank_eof
YC	Yank To a Char	Yank_to_char
YD	Yank Line	Yank_line
YE	Yank To EOS	Yank_window_end
YF	Yank To Finish	Yank_end_line
YH	Yank To Home	Yank_home
YL	Yank Word Left	Yank_word_left
YR	Yank Word Right	Yank_word_right
YS	Yank To Start	Yank_begin_line
YT	Yank To TOF	Yank_tof
YW	Yank Word	Yank_word
ZA	Misc Again	Again
ZB	Misc Back Tab	Back_tab
ZC	Misc Change	Reset_changed
ZD	Misc Date	Date
ZE	Misc Evaluate	Evaluate
ZH	Misc Open Above	Open_above
ZI	Misc InsOvr	Ins_ovr
ZM	Misc Move Right	Move_right

BINDING	DESCRIPTION	NAME
ZN	Misc New Line	Insert_return
ZO	Misc Open Line	Open_below
ZP	Misc Paint Scrn	Paint_screen
ZR	Misc Repeat	Repeat
ZS	Misc Sound	Sound
ZT	Misc Tab	Tab_key
ZU	Misc Undo Delt	Undo
ZX	Misc Execute	Exec_command
ZY	Misc Enter-Auto	Auto_return
ZZ	Misc Enter	Enter_key

Appendix B: Filters

This appendix gives some examples of filters that can be written to work with CoEdit. (See Chapter 7 for description of Block Filter command.)

All filters must take their input from the standard input device (stdin) and write their output to the standard output device (stdout). There need not be a one-to-one correspondence between the number of lines that the filter takes as input and the number of lines that it writes as output. In fact, the filter need not write anything to the standard output device stdout at all.

Example 1

This filter program is written in C; it simply throws away all lines that begin with periods and writes the other lines to standard output.

```
/* This is CoEdit sample filter #1. */
/* The program will read standard input. It
   will throw away all lines beginning with
   a period (".").* It will write the other
   lines to standard output.
*/
#define  MAXLINELEN  255  /* max length */

char  *gets();            /* declaration */

main()

{
  char  s[MAXLINELEN];    /* line buffer */

  while  (gets(s))        /* read stdin */
    if  (*s != '.')       /* if no . */
      printf("%s0, s);  /* write stdout */
}
```

Example 2

This filter program is also written in C; it scans the input lines looking for quotes and then converts to uppercase all characters that appear inside the quotes.

```
/* This is CoEdit sample filter program #2. */

/* The program will read standard input.* It
   will change all letters inside double quotes
   to uppercase.* It will write its output to
   standard output.
*/

#define    MAXLINELEN  255     /* line length */
#define    DOUBLEQUOTE  '

char  *gets();                 /* declaration */

main()

{
  int    inquote = 0;          /* in quote? */
  char  s[MAXLINELEN];         /* line buffer */
  char  *p;                    /* pointer in s */

  while  (gets(s))  {          /* read stdin */
    for  (p = s; *p; p++)  { /* loop thru */
      if  (*p == DOUBLEQUOTE)/* is quote? */
        inquote = !inquote;  /* toggle flag */
      if  (inquote)          /* if in a " */
        *p = toupper(*p);    /* upper it */
    }
    printf("%s0, s);        /* write stdout */
  }
}
```

Appendix C: CoEdit Messages

This appendix lists common CoEdit messages in alphabetical order and the reasons they may be displayed.

DISK IS FULL

CoEdit has received an error from operating system in reading or writing to the disk. Check your disk(ette) condition. It may contain an error, or may simply be full.

A directory is required

The current command is requesting a directory name.

A filename is required

The current command is requesting a file name.

Add to file mode or replace it?

When setting file attributes with the Xternal Chmod command, you may replace the existing file attributes or add to them. See the Xternal Chmod command in Chapter 24.

Bad base - 2, 8, 10, or 16

The Environment Configure command lets you set the default base for CoEdit's expression evaluator. Only these bases are valid. See Environment Configure in Chapter 10 and the Miscellaneous Evaluate command in Chapter 26.

Bad disk selection

The disk you have entered does not exist. File name format is

/[*path*]/*filename*[.*extension*]

Bad macro definition format

To compile a macro, the source must be in the current window. It must also have two special lines at the top of the file, indicating the name of the macro and a comment. Each of the first two lines must begin with an asterisk (*). (See the Macro Compile command in Chapter 17.)

Bad NEXT

The Verify Syntax command has found an unmatched NEXT statement in your BASIC source file. Every NEXT must have a preceding FOR.

Bad or invalid line number

The Goto Line command is requesting the number of a line in the current window. You have entered something that is not a number or is larger than the number of lines in the current window.

Bad WEND

The Verify Syntax command has found an unmatched WEND statement in your BASIC source file. Every WEND must have a preceding WHILE.

Block undefined or out of order

A block in CoEdit is defined by the begin and end block markers. These are set with the Block Begin and Block End commands. All Block operations like move, copy, print, save, etc. require that both markers be set and that the begin marker is before the end marker in the window. (See Chapter 7 concerning Block Commands.)

Cannot locate key table file

CoEdit requires the file co.tab to be in the master directory for all macro compilations and decompilations. This file is provided on the CoEdit distribution diskettes. Copy it into the master directory if you wish to enter or see macros in source form.

Cannot make directory

The operating system has reported an error in attempting to make the specified directory. It, or a file with its name, most likely already exist.

Cannot open error file

After a Verify Compile command, the error file generated by the compile could not be opened. CoEdit constructs the name of this file using the Error Directory option of the Environment Configure command and the Error Extension option of the current language, as defined with the Environment Language command. If the error directory is "/usr/source/err", the file compiled is "TEST.ASM", and the error extension specified for Assembly files is "LST", then CoEdit will search for "/usr/source/err/test.lst" in order to read in errors generated by your assembler. (See the Environment Configure, Environment Language, and Verify Compile commands.)

Cannot perform operation from within block

Operations like Block Move and Block Copy cannot (for your protection) be executed when the cursor is within the current block. Move the cursor outside the current block to perform these commands. See Block commands in Chapter 7 and the Goto Begin and Goto End commands in Chapter 12.

Cannot read filter output

CoEdit uses temporary files when it performs a filter operation on your text. For some reason, CoEdit was unable to read one of these files. Verify that your disk is not out of space.

Cannot remove that directory

The operating system will not let you remove a directory with files in it. Use the System Delete command to delete files in the directory, then try again.

Cannot replace on more than one line

CoEdit's Locate and Replace command cannot currently replace text that spans more than one line. Often a macro can be used to repeatedly find such text, then delete it.

Cannot while printing

You cannot configure your printers while they are active. To cancel printing, use the Print Terminate command.

Character not found

The Delete and Yank to Character commands look for the specified character on the current line. This character was not found.

COPY FAILURE ON FILE

The System Copy command got an error return code while copying the named file. The command has terminated with remaining files (if any) not copied. You may be out of disk space.

Could not execute command

CoEdit attempted to execute the requested command but received an error return code from the operating system. This may occur due to memory limitations, but does not mean the command was not found on the disk.

Directory exists

The specified directory already exists; you cannot make it.

Directory not allowed

The current command is requesting a file name, not a directory. To see the files in a directory, enter the directory name and press ^F. Use the arrow keys and PgUp, PgDn to move through the list of names. ^S will select the highlighted name.

Disk file not compatible

CoEdit has encountered lines longer than 254 characters or binary characters in this file. CoEdit does not currently edit these files.

End of statement not found

The Verify Syntax command cannot find the end of a C or PASCAL language statement.

File cannot be wild

The current command is requesting a single file only, so the wildcard characters asterisk and question mark are illegal.

File does not exist

The specified file does not exist in the specified directory.

File exists

The specified file already exists.

File has been altered. Ok to quit?

You are about to quit a window containing edited material. If you respond Yes, the editing will be lost. If you respond No, the command will be canceled. See the Environment Configure command, option Safety prompts to turn this type of caution off.

Files must both be wild or not wild

The System Rename and Copy commands require matching source and target file names. If the source file specification includes wildcards, so must the target.

Filter not responding

CoEdit is unable to read the results of a Block Filter or Block Program operation.

Help not available

CoEdit cannot locate its help file in the master directory. This file is supplied on the CoEdit distribution diskettes and contains the on-line manual.

Hold files?

The Print commands will let you hold files put on the print queue for later startup, in case you need to set up your printer or want to do some further editing.

Improper), (, }, {

Incomplete statement

These messages are displayed by the Verify Syntax command when it detects these errors in your source file. See the Goto Match command.

Invalid buffer choice

The Internal Del Buffer command lets you flush one or all of your delete buffers from memory. A list is presented, with buffers numbered from 1. Enter a number on the list or 'A' for all buffers.

Invalid column number

The Goto Column command requires a number from 1 to 255, the maximum line length in CoEdit.

Invalid display adapter

The Internal Snow Check command will toggle CoEdit's algorithm for displaying characters on certain color monitors such that snow is or is not shown. This command is not valid for monochrome monitors and adapters.

Invalid pattern

CoEdit patterns use a modified regular expression format. Certain characters like asterisk, backslash, and question mark have special meanings in search and replace strings. See Chapter 16 concerning Locate Commands, for a description and examples of patterns.

Invalid range. Format is LOW-HIGH

Your input for this option must be between LOW and HIGH.

Invalid window choice

You may specify windows by their number or short name. See the Window Assign command for details on assigning a short name.

Language not defined

CoEdit cannot determine what language your source file is in. The default file extensions for each language are in Chapter VERIFY, Verify Commands. If you have a compiler that requires an unusual extension, you may have to use a batch file in the Environment Language command option for this language and insert operating system copy commands to copy a source file with an extension that CoEdit is expecting to one that your compiler will read.

Line would be too long

CoEdit has a maximum line length of 255 characters. The insertion or replace operation would have created a line longer than this. Or, you are trying to center too much text within the current margins.

Another way you might encounter this message is when using the Window Format Command. EBCDIC files have a line length limit of 80 characters.

Macro calls nested too deep

CoEdit has no more memory for nested macro calls.

Macro file missing

If a macro file is deleted or renamed after it has been executed, CoEdit will be unable to locate it.

Marker not set

You cannot Goto Text Marker to this marker, because it was not set with the Text Set Marker command. The markers that are set are displayed in the Goto Text Marker prompt.

More files to edit

A File Open or File Read command has initiated multiple file processing and there are more files to be edited. To cancel the Quit command, enter "N" (No).

Nesting too deep

The Verify Syntax command has a limit to the number of nesting levels it will track. Your source file has exceeded this limit and the command is canceled.

No begin block marker set

CoEdit Block operations require both begin and end block markers to be set with the Block Begin and Block End commands (see Chapter 7).

You will also get this message if you use the Goto Begin Blk command and the begin block marker is not set.

No compiler command for this language

You must set the command to be used to compile source files in this language. CoEdit uses the file extension to determine which language your source is in. Use the Environment Language command to specify the command to be used to compile this language.

No configuration file found

CoEdit's configuration files could not be found in the master directory or the current directory.

No delete buffers exist

There is currently no deleted text. You may get this message using the Miscellaneous Undo, Delete Undo, Delete Yank, or the Internal Del Buffer command.

No end block marker set

CoEdit Block operations require both begin and end block markers to be set with the Block Begin and Block End commands (see Chapter 7).

You will also get this message if you use the Goto End Blk command and the end block marker is not set.

No more files to edit

The File Another or File Next commands cannot find any more files in the pattern specified by the last File Open or File Read command. All matching files have been processed.

No previous locate command

The Locate Again command repeats the last Locate command. When the program is started, there is no Locate command defined.

No such directory

The specified directory cannot be found.

No window has a block defined

The Window Copy and Move commands require another window to have a block defined. No non-current windows have blocks.

No windows are hidden

The Window Visible command will make hidden windows visible, but there are no windows hidden.

No windows with delete buffers

The Window Undo command will insert deleted text from one window into the current window, but no other windows have deleted buffers.

Not in macro define mode

You cannot use the Macro End Def command unless you are currently defining a macro.

Odd ' or Odd "

The Verify Syntax command has detected uneven quotes in your source.

Press Ins to set, Del to reset, Space to toggle

The Window Tab and Environment Tab commands allow you to set hard tabs at tab stops by moving the cursor on a line 255 characters long, folded onto five screen lines. To set a tab, press Ins or space. To delete a tab, press Del or space when the cursor is on it.

Press PgDn, End or ABORT keys

Press PgUp or PgDn keys

Press the PgUp, PgDn, or ABORT keys

CoEdit has multiple screens of information to display. You may use the PgUp and PgDn keys as indicated to scroll through the text, or the ABORT key to stop viewing the information.

Print queue is full

There are either too many files in the print queue or CoEdit has no more memory to add files to the print queue.

Print queue is not empty

CoEdit will display this message if you attempt to leave CoEdit while the print queue is not empty. Use the Print Terminate command to clear the queue.

Replace patterns must match. ^H for help

See the description of the Locate Replace command in Chapter 16.

Replace? Yes, No, All quick, ^Interrupt

You may replace the occurance highlighted on the screen (Yes), skip this replacement (No), automatically replace this and all following occurances (All), or Interrupt the Locate Replace command.

Snow checking off or Snow checking on

The Internal Snow Check command will adjust CoEdit's algorithm for displaying characters on certain color monitors to either allow or eliminate snow.

Starting line must be greater than ending line

The starting and ending lines given to System Type must be in the correct order.

Syntax error in program

The Verify Syntax command has detected a syntax error in your C or Pascal souce file.

That file exists. Overwrite it?

Many CoEdit commands write data to the disk. The file name you have specified already exists and CoEdit wishes to verify that you wish to overwrite the existing file. This and other cautionary messages can be turned off with the Environment Configure Safety Prompts option.

That key is already defined

When defining a non-temporary macro, you have specified a key binding that is already taken.

The print queue is empty

You have asked to see the print queue, but it is empty.

Unmatched FOR/WHILE

The Verify Syntax command has detected an error in your BASIC source. Each FOR statement must have a matching NEXT and each WHILE must have a matching WEND.

Unopened comment terminated

Unterminated comment

The Verify Syntax command has detected a syntax error in your source file.

Warning! File is already loaded

If you load a file into a window that is already loaded, CoEdit displays this message and makes the new copy of the file Read Only, which means you cannot modify it. If you wish to do so, you must use the Window Read Only command to change the read/write status of the window.

Window(s) have been changed

CoEdit will display this message if you attempt to leave CoEdit and there are edited files not saved to the disk. Use the File commands to save these files to the disk. This and other cautionary messages can be turned off with the Environment Configure Safety Prompts option.

Appendix D: Stream Editing with CoEdit

This appendix describes stream editing, which is the editing of data in batch processes. (Stream editors have usually been line-oriented editors found on mainframes and minicomputers.)

Using the -A Command Line Switch

To provide CoEdit with the instructions necessary for stream editing, you need to invoke CoEdit with the -A (Automatic macro) command line switch. This switch tells CoEdit that immediately upon loading, it must begin execution of the macro specified by the -A switch. The macro you specify must contain the CoEdit instructions to transform the file you specify (on the command line). For example, you might say

```
CoEdit -A:massage datafile.out
```

This command line would tell CoEdit to load the file datafile.out and begin execution of the macro called massage. Presumably, the macro will terminate CoEdit so that the operating system batch file containing this line can continue.

Suppose, for example, that you have a program that will analyze files on your disk and as its output generates a list of files that need some kind of updating. You might feed this list into your stream editor. The output from the editor would be those file names with some command in front of each one (for example, compile or sort), thus generating an executable batch file or script. Then you can execute the batch file. Here is a batch file that might accomplish such a task:

```
 analyze > datafile.out
 co -A:NAMELIST datafile.out
chmod +x process
 process
```

This example assumes the following:

- there is a macro named NAMELIST.cmf
- the macro will terminate CoEdit
- the macro will instruct CoEdit to write the resulting data to the file "process".

Editing Multiple Files

Stream processing can also perform complex editing on many files. If you have a program generate a list of file names to be edited into a file named MODS.LST, as in the following example:

```
plist.c
pout.c
pform.c
pname.c
pjoin.c
```

Now suppose you have a macro called DOIT that performs some complex operations on a file, like deleting the first 10 lines and replacing some text patterns:

```
*DOIT
*Prepares the current file and saves it
Repeat "10" Enter Del_line
Locate_replace "*.asm" Enter "/work/*.asm" Enter
"AQ" Enter
File_done
```

To execute the DOIT macro on each file in the analyze list, you need another short macro, STREAM:

```
*STREAM
File_load "MODS.LST" Enter
do while not eof
   Yank file name, open, call DOIT
   Yank_end_line Go_down_line
   File_open CtrlU Enter
   Exec_command "DOIT" Enter
enddo
Quit_exit
```

Now, loading CoEdit with -A:STREAM will cause each file to be opened, changed, and saved. Note the use of ^U at a prompt in STREAM (for the Read from which file?).

You may also set the automatic macro with the Environment Macro command which is described in Chapter 10. This will ask CoEdit to execute this macro each time CoEdit is loaded. However, while you may not want to stream edit each time you load CoEdit, you may want to execute a macro that will set up certain parameters or perform some task each time you load CoEdit.

Index

^A Append default string at prompts, 2-5
^B Blank input field at prompts, 2-5
^D Use default response at prompts, 2-5
^F Show available files at prompts, 2-5
^G Grab text at current word for prompts, 2-6
^L Search for file at prompts, 2-6
^L Searching Method, 4-4
^N Insert current file name at prompts, 2-6
^Q Display print queue at prompts, 2-6
^S Selects highlighted filename at prompts, 2-6
^T Enter control characters at prompts, 2-7
^U Undo delete buffer at prompts, 2-7
^X Transpose last two characters at prompts, 2-7

A
ABORT key
  definition, 2-2
  setting, 1-4
ASCII, 1-5
  character, 13-2
  table of characters, 13-5
Assign windows, 23-5
AutoIndent, 26-19
Automatic Macro Switch, 1-2
Automatic
  file saving, 1-6
  macros, D-1

B
Back Tab, 26-3
BACKSPACE key
  text entry, 2-3

Batch files
processing, 1-2
scripts, 1-2
Block
all, 7-5
begin, 7-6
columnar, 4-8
copy, 7-7
definition, 4-7
delete, 7-9
end, 7-11
filtering, 7-12
for language statements, 7-20
horizontal, 4-8
lower, 7-14
move, 7-15
narrowing, 7-17
program, 7-19
quick, 7-20
remove, 7-22
saving to disk, 7-23
status, 4-11
text wrap, 7-24
upper, 7-26
vertical, 4-8, 7-27
widen, 7-28
word wrapping, 7-24
yank, 7-29
Border
messages, 4-12
window, 4-6
Breaking
macros, 5-2, 17-5
Buffer
definition, 4-7, 4-8
delete, 4-8, 13-3

C
Canceling a command, 2-4

CAPS LOCK key, 2-3
Caution, 4-2
Centering text lines, 21-5
Changed flag, 26-4
Changing
  attribute, 24-2
Character
  ASCII value, 13-2
Checking syntax, 22-11
chmod, 24-2
Clearing border messages, 4-12
Close window, 11-5, 11-17, 19-1
cmf suffix, 5-2, 5-3
CML
  Border_msg, 6-3
  Break, 6-4
  Breakpoint, 6-5
  Cancelmac, 6-6
  Continue, 6-7
  debugging, 6-1
  Do, 6-8
  Endmac, 6-9
  Execmac, 6-10
  Foreach_window, 6-11
  function references, 5-3
  Get_Enter, 6-12
  GetYN, 6-13
  Goto, 6-14
  If, 6-15
  Label, 6-16
  logical values
    list of, 5-8
  macros, A-1
  No_Update, 6-17
  Proc_menu, 6-18
  Prompt, 6-20
  Restart, 6-21
  source format, 5-3
  uncompiling, 5-4
  Update, 6-22

Varfield, 6-23
Wait, 6-24
While, 6-25
cms suffix, 5-3, 5-4, 5-5
coC, 17-7, 17-21
CML compiler, 5-4
error files, 5-5
CoEdit Files
co.cfg, 10-1
co.nfy, 10-1
Colors
setting screen, 10-5
Column number, 4-11
Columnar blocks, 4-8
Command groups, A-1
Command line switch
-A, 1-2
-C, 1-3
-D, 1-3
-F, 1-3
format of, 1-2
-KA, 1-4
-KK, 1-4
-KP, 1-4
-KT, 1-4
-L, 1-5
-O, 1-5
precedence of, 1-6
Command
entry, 4-13
environment, 1-6
file, 4-7
group, 4-1
keys, 4-14
miscellaneous, 26-1
mode, 4-13
names, 4-1
window, 4-7
Communication window, 2-3, 4-7
Comparing windows, 23-7

Compilation, 17-7, 22-3
  examples of scripts, 22-3
Compiler, 17-18
  information, 10-20
  setting the interface for, 22-1
Configuration
  directory, 10-29
  files, 10-1, 10-2
  macro, 10-23
  tabs, 10-31
co.nfy, 10-1
Control characters
  entering into your text, 21-9
Converting
  lowercase, 7-26
  uppercase, 7-14
Copying
  block, 7-29
  block text, 7-7
  text, 7-29
  windows, 23-9
co.tab, 17-18
Creating
  directory, 24-4
  shell, 24-6
Cursor, 12-10, 14-11, 14-12

D

Date
  display, 26-5
  inserting into the text, 21-6
  setting, 24-7
Defining
  language, 10-20
Delete buffers, 4-8, 9-1, 13-3
DELETE key
  text entry, 2-3
Deleting
  block, 7-15

block text, 7-9
Delimiters
definition of, 4-9
Directory, 20-6
attribute, 24-1
changing, 20-4
creating, 24-4
displaying, 2-6
removing, 24-5
working, 20-4
Display
ASCII table, 13-5
character value, 13-2
commands, 15-2
scan code, 13-4
Duplicating
character, 21-8
line above, 21-7

E
Editing
area, 4-9
stream, D-1
Empty menus, 3-2
ENTER key, 26-20
text entry, 2-2
Environment
attribute, 10-5
changing, 10-6
command, 1-6, 2-1
configure, 10-6
definition of, 10-1
language, 10-20
macro, 10-23
printer, 10-27
read, 10-29
saving on the disk, 10-30, 10-36
tabs, 10-31
update, 10-36

ERR, 22-9
Error, 22-7, 22-8, 22-9
Error messages, 4-12, C-1
Escape sequences
  setting timeout for, 1-4
Evaluation
  of expressions, 26-6
Examples
  ^F processing, 11-1
  ^L processing, 11-2
  Block insert, 7-2
  Filter programs, B-1
  Key bindings, 10-3
  macro source, 17-6
  Menu definitions, 10-2
  stream editing, D-2
Execute
  command, 26-18
  macro, 26-18
Executing
  command, 24-3
  program, 24-1
EXIT
  operating system command, 24-3
Exiting CoEdit, 19-1
Expanding windows, 23-10

F
F1 at help screens, 3-2
F1 for help at prompts, 2-6
File Searching Method, 4-4
File
  autosave, 1-7
  commands, 4-7
  copying, 20-2
  definition, 4-4
  deleting, 20-3
  displaying on screen, 20-9
  entering file names, 2-6

exchange, 11-7
list, 20-6
load, 11-9, 11-12
merge, 11-11
moving between directories, 20-4, 20-8
name, 1-1, 11-1
new, 11-4
next, 11-12
open, 11-14
read, 11-15
renaming, 20-8
reset, 22-5
save, 11-5, 11-16
selection, 2-6
startup configuration, 1-6
tag, 22-13
update, 11-17
Filters, 7-12
examples of, B-1
Find all mode, 16-3
Find not all mode, 16-3
Foreign Characters Switch -F, 1-3
Function keys, 4-14

H
Help, 3-1
at prompts, 2-4, 3-2
command menus, 3-1
general, 3-2
on-line manual, 3-2
Hiding windows, 23-12
Horizontal blocks, 4-8

I
Input
^F File Display, 11-1
^L Locate File, 11-2
Insert, 12-5

Insert mode, 2-2, 4-14
Insert/Overtype flag, 26-9
Inserting
  blocks of text, 7-2
  horizontally, 7-2
  text, 7-19, 11-11
  vertically, 7-4
International Characters, 1-3
Interrupting
  macros, 5-2, 17-5

K
Key bindings
  saving on the disk, 10-30, 10-36
Key
  ABORT, 2-2, 2-4
  BACKSPACE, 2-3
  CAPS LOCK, 2-3
  changing the definitions of, 4-14
  commands, 15-2
  defining in co.nfy, 10-2
  DELETE, 2-3
  ENTER, 2-2
  for commands, 4-14
  function, 4-14
  list, 15-3
  redefining, 15-1
  RETURN, 2-2
  Scroll Lock, 4-12
  TAB, 2-3
Keyboard, 2-1, 4-1, 13-4
Keyboard Timeout
  setting, 1-4
Keystroke Typeahead
  setting, 1-4

L
Languages

determining, 22-1
extensions of, 22-1
Leaving CoEdit, 19-1
Line
number, 4-10
reset, 22-6
List windows, 23-13
List
function keys, 15-3
Loading
CoEdit, 1-1
file, 11-4
Lowercase, 7-26

M
Macro
#debug directive, 17-6
automatic, 10-23, D-1
binding to keys, 10-4, 17-5
canceling execution of, 5-2, 17-5
CML, 5-2, A-1
command environment, 2-1
commands, 17-3
compiling, 17-4, 17-6
creating source for, 17-3
debugging, 5-6, 17-5, 17-14, 17-21
defining, 17-9, 17-12
by keyboard, 17-1
disk files, 5-2
editing, 17-4
embedded commands in, 17-3, 17-9
end definition, 17-12
ending keystroke recording, 17-2
executing, 5-5, 17-1
flush, 17-13
from operating system, 5-4
general description, 5-1
go, 17-14
in memory, 5-2

key, 17-15
key bindings, 17-1, 17-11
keys allowed, 5-1
library, 10-23, 17-5
directory, 5-2
list, 17-16
maximum size, 10-24
memory, 17-13
name, 17-10, 17-15
nesting execution, 17-5
sample source, 17-6
saving, 17-11, 17-19
temporary macros, 17-2
symbols, 5-7
temporary, 10-4, 17-2, 17-9, 17-19
tokens, 5-7
tracing, 17-14
execution, 17-21
uncompile, 17-24
user description, 17-10
within CoEdit, 5-4
Markers, 12-2, 12-10
removing, 7-22, 21-10
setting, 21-11
Master directory, 1-3
changing default, 1-3
Menus, 3-1
command groups, 3-1
configuring in co.nfy, 10-2
debugging, 17-21
empty, 3-2
Messages, 4-2
error, 4-12, C-1
locations on screen, 4-12
Miscellaneous commands, 26-1
Moving
block, 7-15
cursor, 26-10
windows, 23-14
Multiple file processing, 11-2

N
Narrowing
window, 7-17
New line, 26-11
Next windows, 23-15
Normal windows
definition of, 4-5

O
On-line
help, 3-1
manual, 3-2
Opening
file, 11-4, 11-14
window, 11-14, 11-15, 23-16
Other Graphics switch, 1-5
Overtype mode, 2-2, 4-14, 12-8

P
Paint screen, 26-13
Parameters
changing CoEdit, 10-6
Patterns in locate commands, 16-4
Prefix key
defined, 2-1
setting, 1-4
Preprocessors, 22-2
Previous windows, 23-17
Print, 10-27, 18-1
block, 18-4
cancel, 18-5, 18-6
file, 18-8
form feed, 18-7
hold, 18-10
number of copies, 18-2
pause, 18-12
queue, 18-6, 18-14
queue displaying, 2-6

release, 18-10, 18-16
terminate, 18-17
window, 18-18
Prompts, 4-2
default strings, 2-4
general, 2-3
help, 2-4
help at, 3-2
input at, 2-3
leaving, 2-4
special keys at, 2-5
Yes/No, 2-4

Q
Quit, 19-1
save, 19-2
terminate, 19-4
windows, 19-6, 23-18

R
Read
configuration, 10-29
Read-only windows, 23-19
Redefining keys, 15-1
Regular expressions, 16-4
Removing text
line, 25-5
to a character, 25-4
to bottom of file, 25-3
to end of screen, 25-6
to finish of line, 25-7
to home, 25-8
to left, 25-9
to right, 25-10
to start of line, 25-11
to top of file, 25-12
word, 25-13
Removing

directory, 24-5
markers, 7-22, 21-10
Repainting the screen, 26-13
Repeating commands, 26-2, 26-14
Resetting
file, 22-5
line, 22-6
markers, 7-22
Restrictions, 4-2
description of, 4-2
RETURN key
text entry, 2-2

S
Safety
Warning prompts, 11-5, 11-7, 11-9, 11-16, 11-17, 19-2, 19-4, 19-6, 23-18
Save File, 1-7
Saving
block text, 7-23
environment, 10-30
file, 11-5
window, 11-12, 11-17
Screen
attributes
changing, 10-5
columns
setting, 1-3
layout, 4-9
lines
setting, 1-5
updating
affected by typeahead, 1-4
Scripts, 1-2
examples for compilation, 22-3
Scroll, 14-3, 14-10
Scroll Lock
key, 4-12
processing, 4-7

Setting
block markers, 7-5, 7-6, 7-11, 7-20
CoEdit Parameters, 1-6
configuration, 10-6
date and time, 24-7
language, 10-20
macro, 10-23
markers, 21-11
printer, 10-27
screen attributes, 10-5
tabs, 10-31
Shell
creating, 24-1
invoking, 24-6
Shift state, 2-3
Short name for windows, 23-2
Size
windows, 23-20
Slide windows, 23-25
Sound, 26-15
Special keys
ABORT, 2-2
ABORT to previous prompt, 2-4
BACKSPACE, 2-3
CAPS LOCK, 2-3
DELETE, 2-3
ENTER, 2-2
for commands, 4-14
RETURN, 2-2
Scroll Lock, 4-12
TAB, 2-3
Status line, 4-9, 4-10
Stream editing, 1-2, D-1
Subwindows, 4-5
Switch
-A, 1-2
-C, 1-3
-D, 1-3
-F, 1-3
-KA, 1-4

-KK, 1-4
-KP, 1-4
-KT, 1-4
-L, 1-5
-O, 1-5
Symbols
for macros, 5-7
Syntax check, 22-10

T
TAB key
text entry, 2-3
Tabs, 2-3, 21-4, 23-21, 26-16
Fixed, 21-3
guide indicator, 21-3
Hard, 21-2
setting, 10-31
Soft, 21-1
Tag, 22-13
Terminal keyboard problems
solving, 1-4
Text entry, 4-8, 4-11, 4-14
definition, 4-14
mode, 4-14
special keys, 2-2
Text
centering, 21-5
copying, 7-29
deleting, 9-3
duplicating, 21-7, 21-8
inserting, 11-11, 12-5
marker, 12-10
removing, 21-10
set, 21-11
merging, 11-11
moving, 14-6, 14-7
overtype, 12-8
saving to disk, 7-23
tab, 21-4

wrapping, 7-24
Time
display, 26-5
inserting into the text, 21-6
setting, 24-7
Tokens
for macros, 5-7
Total lines, 4-11
Typeahead
controlling, 1-4

U
Uncompiling, 5-4
Undo
commands, 4-8
definitions of, 4-9
delete, 26-17
deleted text, 4-9
find all level, 16-3
find all subwindows, 4-9
of deletions, 9-1, 25-1
windows, 23-22
UNNAMED windows, 4-7
Uppercase, 7-14

V
Vertical blocks, 4-8, 7-27
Visible windows, 23-24

W
Warning prompts, 11-5, 11-7, 11-9, 11-16, 11-17, 19-2, 19-4, 19-6, 23-18
Widening
of block, 7-28
Windows, 4-5
assign, 23-5
attributes, 23-2

border, 4-5, 4-6
bury, 23-6
changed, 19-6
close, 19-1
commands, 4-7
compare, 23-7
copy, 23-9
definition, 4-4
discard, 11-9
expand, 23-10
hiding, 23-12
introduction, 23-1
list, 12-11
listing, 23-13
moving, 23-14
moving between, 23-2
next, 23-15
open, 4-11, 11-15, 23-16
opening and closing, 23-1
previous, 23-17
quit, 23-18
read-only, 23-19
save, 11-16, 11-17, 19-1, 22-3
short name, 23-2
size, 23-20
slide, 23-25
tab, 23-21
undo, 23-22
visible, 23-24
Working directory, 20-4
Wrapping
text, 7-24

X
Xternal
chmod, 24-2
commands, 24-2
directory, 24-4, 24-5
execute, 24-3

shell, 24-6
touch, 24-7

Y
Yank
block, 7-29
line, 25-5
to a character, 25-4
to bottom of file, 25-3
to end of screen, 25-6
to finish of line, 25-7
to home, 25-8
to start of line, 25-11
to top of file, 25-12
word, 25-13
word left, 25-9
word right, 25-10

INTERACTIVE

A Kodak Company

DOC0112-2Z